

PICOSTRAIN

临时手册

PS081 衡器系统单芯片方案

2010 年8 月13日

文件名称: DB_PS081_cn V0.8



Published by acam-messelectronic gmbh

© acam-messelectronic gmbh 2010

Disclaimer / Notes

„Preliminary“ product information describes a product which is not in full production so that full information about the product is not available yet. Therefore, acam messelectronic GmbH („acam“) reserves the right to modify this product without notice. The information provided by this data sheet is believed to be accurate and reliable. However, no responsibility is assumed by acam for its use, nor for any infringements of patents or other rights of third parties that may result from its use. The information is subject to change without notice and is provided „as is“ without warranty of any kind (expressed or implied). Picostrain is a registered trademark of acam. All other brand and product names in this document are trademarks or service marks of their respective owners.

Support

For a complete listing of Direct Sales, Distributor and Sales Representative contacts, visit the acam web site at: <http://www.acam.de/company/distributors> or refer to chapter 7.2 in this datasheet

For technical support you can contact the acam support team in the headquarter in Germany or the Distributor in your country. The contact details of acam in Germany are:

support@acam.de or by phone +49-7244-74190.

目录

页面

1 系统前瞻	1-2
2 特性和规格	2-2
3 高端转换器	3-2
3.1 概述	3-2
3.2 测量原理	3-2
3.3 连接金属应变传感器	3-3
3.4 放电电容、周期时间、求平均	3-10
3.5 模式和时序	3-15
3.6 预处理	3-25
4 时钟元件 & 特殊设置	4-2
4.1 晶振	4-2
4.2 内部LCD驱动	4-3
4.3 对于外部LCD驱动的支持	4-14
4.4 I/O 管脚	4-16
4.5 SPI 接口	4-19
4.6 电源电压	4-25
5 配置寄存器	5-2
6 中央处理单元 (CPU)	6-2
6.1 结构图	6-2
6.2 存储器结构	6-2
6.3 状态寄存器和结果寄存器	6-5
6.4 指令集	6-8
6.5 系统Reset, 休眠模式和自动配置	6-34
7 其它	7-2
7.1 从 PS08 过渡到 PS081	7-2
7.2 Bug报告	7-3
7.3 文献指导	7-3
7.4 文件历史	7-4
8 索引	8-1
9 附录	9-2

目录	页面
1 系统前瞻	1-2
1.1 特性.....	1-2
1.2 优点.....	1-2
1.3 应用	1-2
1.4 概述.....	1-2
1.5 功能方框原理图	1-3

1 系统前瞻

1.1 特性

- RMS 噪声:
 - 20.1nV快速模式, 5 Hz
 - 11.5nV SINC3滤波开启, 5 Hz
 - 8.9nV SINC5滤波开启, 5 Hz
- 在衡器应用当中高达25万峰-峰值显示分度 (2 mV/V 应变灵敏度)
- 可以灵活设置输出频率从 < 1 Hz到 1000Hz
- 功耗范围: PS081功耗本身仅为
 - $\sim 0.39\text{mA}$ (在最高速度情况下)
 - $\sim 0.005\text{mA}$ (低功耗配置情况下)
 - $\sim 0.001\text{mA}$ 静态电流
- 供电电压范围: 2.1V到3.6V
- 转换器类型: 时间数字转换器 (TDC)
- 测量精度: 28 位 ENOB (RMS)或者 25.8 位 无失码噪声 (峰-峰)
- 24-位内部单片机处理器带有2K可擦写存储EEPROM
- 内部LCD驱动, 可直接驱动 4x14, 3x15 和2x16段
- 驱动外部LCD驱动芯片的端口
- 4-线串行SPI接口
- 内部超低嵌入式10kHz振荡器
- 6个I/O口, 最多可以配置为21输入或者5个输出
- 非常高的电源抑制比 (PSRR)
- 超低温度增益偏移和零点offset偏移
- 内部嵌入的驱动LCD的充电泵
- 内部嵌入的电压带隙基准参考用于电池低压检测
- 看门狗定时器

1.2 优点

- 衡器单芯片解决方案
- 转换器, 单片机和LCD控制器集成在一颗芯片
- 超低系统整体电流消耗 (包括传感器在内降至最低15 μA)
- 超低传感器自热现象
- 软件校正传感器的温度增益和零点offset偏移
- 可提供裸片 (115 μm pitch) 或者封装片 (QFN56, 7x7 mm^2)

1.3 应用

工业

- 扭矩传感器
- 压力显示装置
- 标准商业用秤
- 高端计价秤

消费

- 纯太阳能电池板驱动衡器
- 人体秤
- 厨房秤
- 口袋秤
- 吊秤
- 邮局秤
- 包裹秤

1.4 概述

PS081是一款完美适合超低功耗以及高精度应用的片上系统方案。此芯片是专门为衡器应用所设计, 当然同样也适用于基于金属应变测量的扭矩测量以及其他力学测量应用。此芯片将PICOSTRAIN测量原理的优势完美的进行诠释。另外芯片还同时集成一颗28位的转

换器以及一个24位的内部处理器。另外芯片还集成了LCD驱动，带有很多强大指令和函数的3K ROM，以及可无限次编程的2K EEPROM，内部带有10kHz晶振给芯片提供时钟源。仅需很少一些外部原件就可以建立起完整的称重衡器电路。

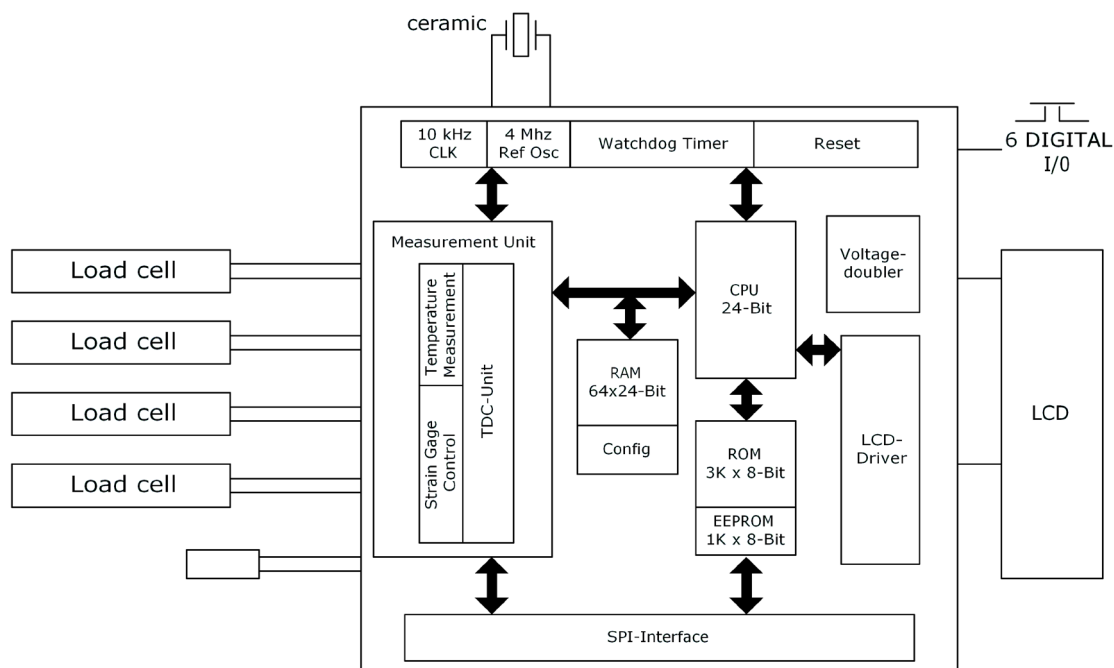
系统的供电电压范围从2.1V到3.6V功耗超低。根据配置不同电流消耗从大约0.005mA到0.4mA不等。芯片输出频率可以灵活调整从 < 1 Hz 最高到1000 Hz。转换内码可高达1百万分度i (28位ENOB RMS) 这样的精度在现今的转换器中是绝对处于高端水平的。这样的高精度是完全与现今最高端的 $\Sigma-\Delta$ -AD 数模转换器相媲美的，而PS081的测量功耗则要低很多。

由于有这些杰出的特性，PS081可以建立一系列多样化的衡器应用电路。在高精度方面它可以实现25万分之一的峰峰值稳定显示分度 (2mV/V传感器)! 另一方面，复杂的电源管理方案和PICOSTRAIN的测量原理特性可以使整个系统的电流消耗包括传感器在内降低到15 μ A! 由于功耗超低，第一次实现了真正不带电池的太阳能驱动衡器! 当然，这些优势可以进行结合，例如将高精度和低功耗结合起来做一款满足商业秤标准的衡器，用2xAA干电池可以工作超过1500小时。

一些其他非常好的特性如通过软件补偿传感器的温度offset和gain偏移，或者可以驱动仅一个半桥或者4个全桥等，使PS081给革新产品开辟了一扇创新的大门。

1.5 功能方框原理图

图 1.1 功能方框原理图



目录	页面
2 特性和规格	2-2
2.1 绝对最大额定值	2-2
2.2 标准操作条件	2-2
2.3 电器特性.....	2-2
2.4 转换器的测量精度	2-3
2.5 积分非线性度	2-4
2.6 精度 vs. 供电电压.....	2-6
2.7 电流消耗.....	2-7
2.8 时序.....	2-7
2.9 管脚分配.....	2-8

2 特性和规格

2.1 绝对最大额定值

表 2.1 绝对最大额定值

符号	参数	条件	最小	最大	单位
V _{cc} V _{cc_load} V _{cc_osc} V _{cc_LCD}	供电电压	V _{cc} vs. GND	-0.5	5.0	V
V _{in}	DC 输入电压		-0.5	V _{cc} + 0.5	V
ESD Rating	FICDM	All pins	2		kV
T _{stg}	储藏温度	塑料封装	-55	150	° C

2.2 标准操作条件

表 2.2 标准操作条件

符号	参数	条件	最小	最大	单位
V _{cc} V _{cc_load} V _{cc_osc} V _{cc_LCD}	供电电压	V _{cc} vs. GND (* 不带LCD, EEPROM 编程以及电压测量的时候)	2.1 (1.5*)	3.6**	V
V _{in}	DC 输入电压		0.0	V _{cc}	V
V _{out}	输出电压		0.0	V _{dd}	V
Top	操作温度		-40	125	° C
T _{stg}	存储温度	塑料封装	-55	150	° C

** 4.5 V for highest resolution within limited temperature range

2.3 电器特性

表 2.3 电器特性

符号	参数	条件	最小	典型	最大	单位
V _{il}	输入低电压	CMOS			0.3V _{cc}	V
V _{ih}	输入高电压	CMOS	0.7V _{cc}			
V _{hyst}	输入滞后	V _{cc} = 3.6 V V _{cc} = 3.0 V V _{cc} = 2.7 V V _{cc} = 2.2 V V _{cc} = 1.8 V		400 280 225 150 80		mV
V _{oh}	输出高电压		0.8			V
V _{ol}	输出低电压				0.2V _{cc}	V
V _{lbat}	低电池电压检测		2.2		2.9	V
LCD_COM LCD_SEG	LCD驱动稳压	lcd_vlt = 0 lcd_vlt = 1 lcd_vlt = 2		2.0 2.5 3.0		V
I _q	静态电流	没有晶振, 没有TDC开启		150		nA
I _{osc}	4 MHz晶振持续工作电流	V _{cc} = 3.6 V V _{cc} = 3.0 V V _{cc} = 2.1 V		200 130 65		µA

2.4 转换器的测量精度

表 2.4 在 $V_{CC} = 3.3V$ 时应用外部比较器情况下的测量性能

	ENOB dR/R 应变电阻		
频率 (Hz)	无滤波	SINC3	SINC5
500	23.8	24.8	25.2
250	24.4	25.2	25.7
100	25.2	25.8	26.1
50	25.5	26.2	26.5
20	26.0	26.8	27.0
10	26.6	27.4	27.7
5	27.2	27.9	28.3

表 2.5 在 $V_{CC} = 3.3V$ 时应用外部比较器情况下的测量性能, 相对于 2 mV/V 应变 (秤重衡器)

	测量精度 @ 2 mV/V 最大输出, 快速模式*			
频率 (Hz)	ENOB	分度 有效值	噪声 nV rms	噪声 nV 峰-峰值
500	14.8	28,000	231	1,386
250	15.4	44,000	148	891
100	16.2	74,000	89	535
50	16.5	95,000	69	416
20	17.0	133,000	49	297
10	17.6	200,000	33	198
5	18.2	294,000	22	135

* 快速模式 = 不带滤波

表 2.6 在 $V_{CC} = 3.3V$ 时应用外部比较器情况下的测量性能, 相对于 2 mV/V 应变 (秤重衡器)
带有 SINC3 和 SINC5 滤波 (滑动平均3次和5次)

	测量精度 @ 2 mV/V 最大输出, SINC3 滤波				测量精度 @ 2 mV/V 最大输出, SINC5 滤波			
频率 (Hz)	ENOB	分度 有效	噪声 nV rms	噪声 nV 峰-峰值	ENOB	分度 有效	噪声 nV rms	噪声 nV 峰-峰值
500	15.8	55,000	118	713	16.2	74,000	89	535
250	16.2	74,000	89	535	16.7	105,000	62	376
100	16.8	114,000	57	347	17.1	142,000	46	277
50	17.2	153,000	42	257	17.5	181,000	36	218
20	17.8	222,000	29	178	18.0	266,000	24	149
10	18.4	344,000	19	115	18.7	416,000	15	95
5	18.9	476,000	13	83	19.3	625,000	10	63

表 2.7 在 $V_{cc} = 3.3V$ 时应用内部比较器情况下的测量性能

频率 (Hz)	ENOB dR/R 应变电阻			ENOB 2mV/V, 快速模式
	无滤波	SINC3	SINC5	
500	23.0	24.0	24.4	14.0
250	23.6	24.4	25.1	14.6
100	24.4	25.0	25.3	15.4
50	24.7	25.4	25.7	15.7
20	25.2	26.0	26.2	16.2
10	25.8	26.6	26.9	16.8
5	26.4	27.1	27.5	17.4

* * 快速模式 =不帶滤波

表 2.8 整体参数

符号	参数	条件	最小	典型	最大	单位
INL	应变积分非线性度	Supply Voltage 3.0V to 3.6V		0.01*		$\mu V/V$
	Offset漂移	整个系统, 350 Ω SG, 3V Wheatstone		10		nV/K
	增益漂移 在 $-20^{\circ}C \cdots +70^{\circ}C$ 范围	半桥** 整个系统, 350 Ω SG, 5V		<1 ~1		nV/K ppm/K
PSSR	V_{cc} 电源抑制比	1.8V 或者 3.3 V ± 0.3 V	106 @1.8V	130 @3.3V		dB

* 相当于 A/D转换器带PGA设置为128时的 ± 1.25 ppm

** 如果应用推荐电路

2.5 积分非线性度

PS081的积分非线性度（INL）可以达到 $\pm 0.01\mu V$ (1.25 ppm)。如果换算为分度值的话这相当于 $\pm 1:20$ 万。这个性能与现今最优秀的A/D数模转换器相比较也是相当出色的。一般的A/D数模转换线性在 $\pm 0.12 \mu V/V$ (15ppm)的范围之内，较好的A/D数模转换器线性度也只不过可以达到 $\pm 0.4\mu V/V$ (5ppm)而已。

当然，如果您的传感器足够准确另外线性足够好的话，您可以用来测试来看电路本身的测量线性度。我们在测试电路部分的线性度的时候应用了acam公司的修订版的传感器模拟器 (ALCS350-V2)，这个模拟器提供了一个非常方便的方式来测试电路部分的非线性，当然也可以用来测试电路部分在温度和电压不同情况下的性能。ALCS350-V2本身的线性在PICOSTRAIN工作方式的情况下为 $\pm 0.01 \mu V/V$ (1:20万)，而在惠斯通工作方式情况下为 $\pm 0.04\mu V/V$ 。这个模拟器的线性要比普通传感器高很多。应用ALCS350-V2 作为传感器来测量PS081的线性表格如下图所示。更多的关于传感器模拟器的性能以及限制请参看我们的ALCS350-V2数据手册。

图 2.1 不同电压情况下的非线性

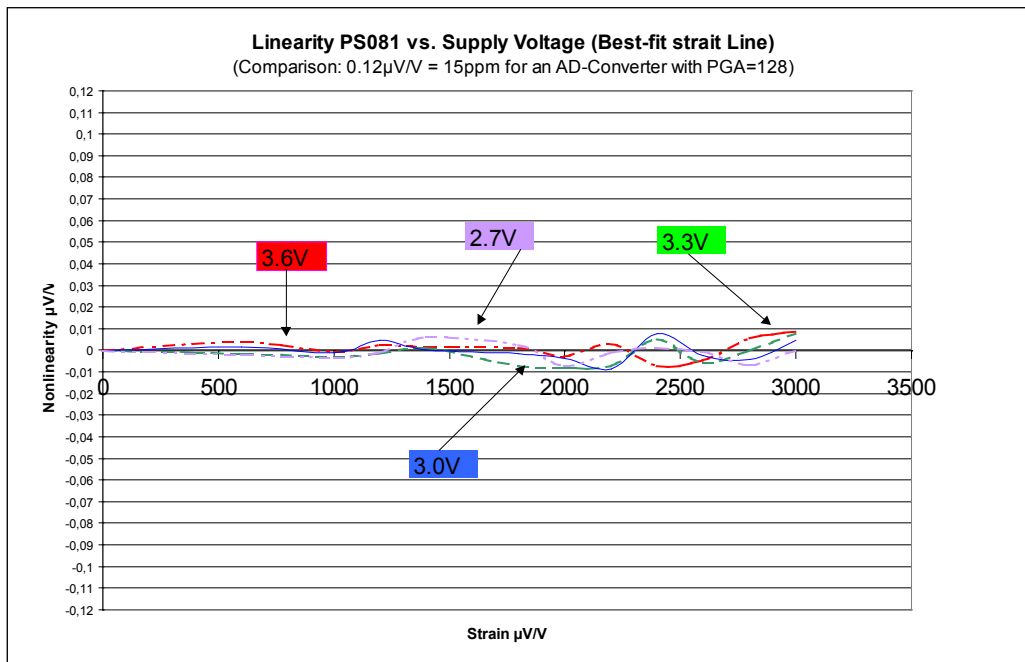
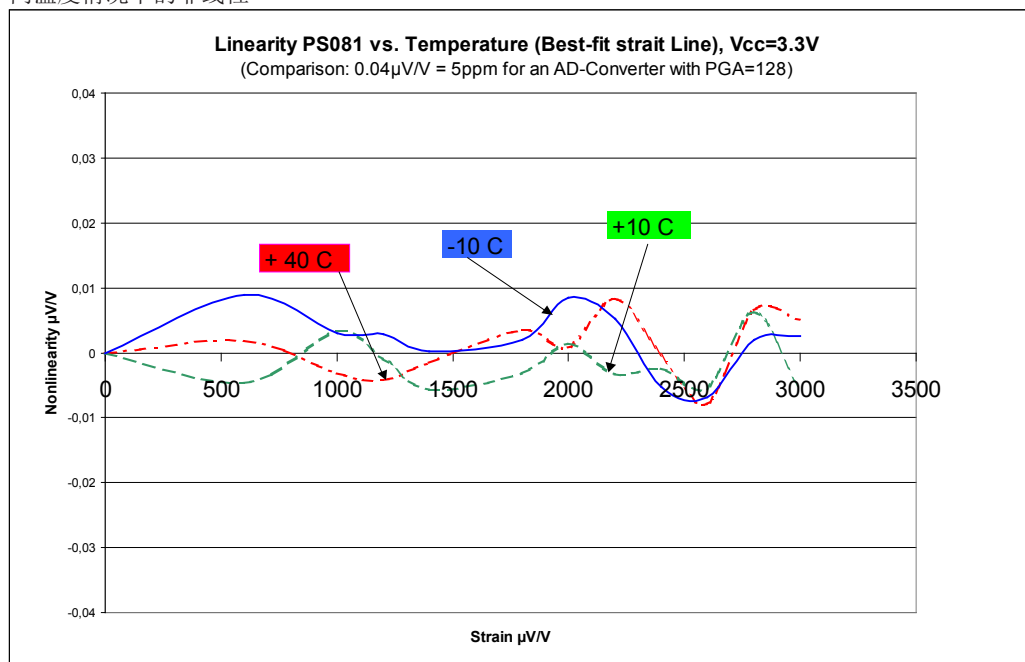


图 2.2 不同温度情况下的非线性



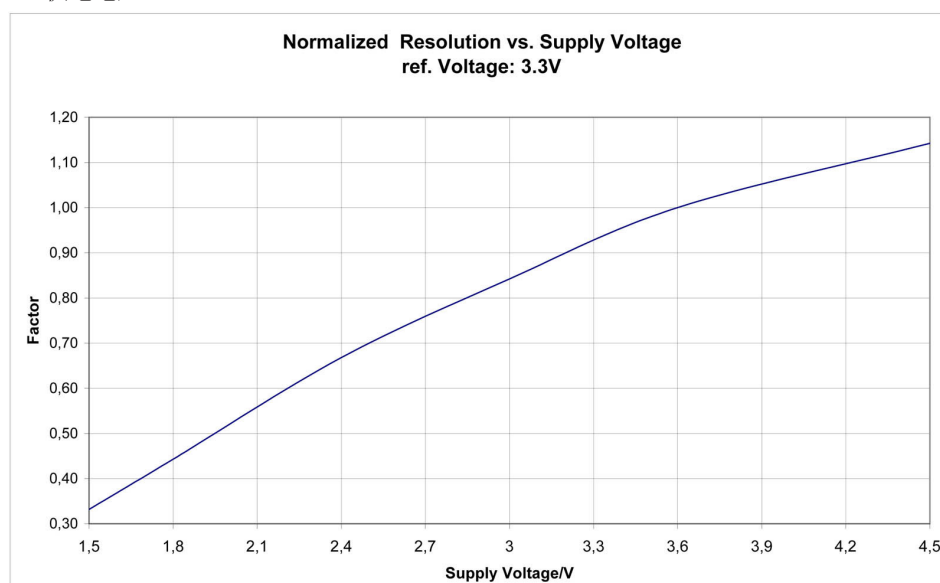
注：

最好的线性是在电压范围3.0V到3.6 V的情况下获得的。线性与所设置的周期时间有关。上面的测量是在周期时间设置为 $\text{cytime}=85$ (等于 $170\mu\text{s}$) 情况下获得的。

2.6 精度 vs. 供电电压

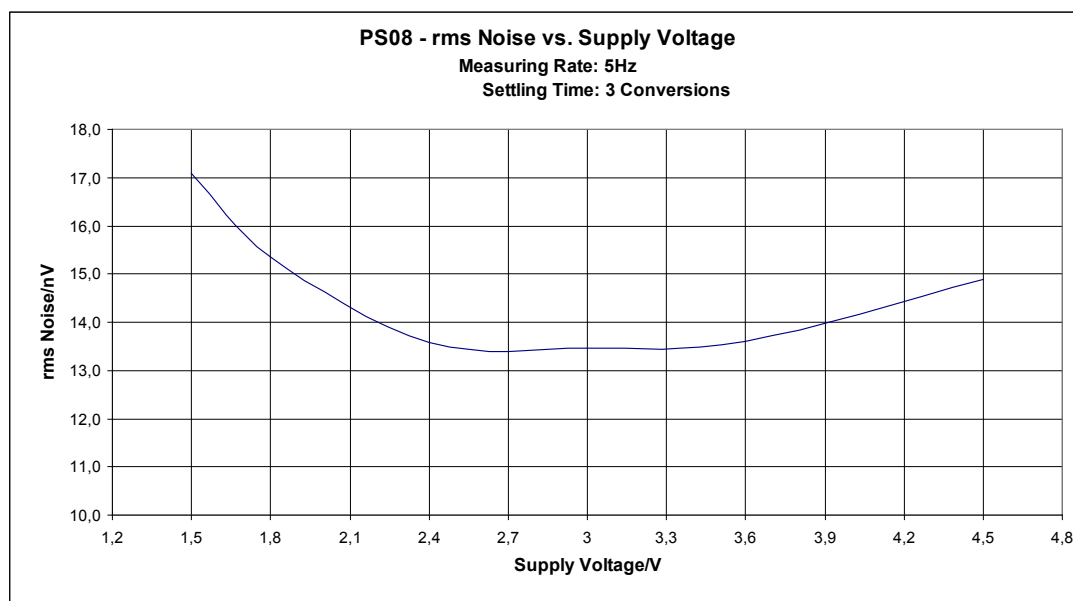
PS081可以在一个非常大的供电范围内工作。那么可以达到的精度与供电电压有直接关系。电压越高，那么所达到的精度就越高。下面的图说明了供电电压与精度的一个关系，所有数值与3V电压为参考。

图 2.3 精度 vs. 供电电压



下面面的图说明了供电电压与噪声的相关性。最低输入噪声在电压为2.4V和3.6V之间获得。最大差分输入电压(例如6.6mV@2mV/V和3.3V供电电压)除以输入噪声给出了有效精度。

图 2.4 RMS噪声 vs. 供电电压



2.7 电流消耗

下面的表格说明了衡器上包括所有电路器件的电流消耗(包括传感器电流)

表 2.9 电流消耗 不同测量精度的情况

分度值 *	刷新率	灵敏度*	在 3V时的操作电流		衡器类型	工作时间
2,000	3 Hz	1 mV/V	1 kOhm	15 μ A	太阳能驱动衡器	
2,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	30 μ A 70 μ A	邮局称, 体重秤, 厨房秤, 口袋秤。	3,000 小时 (1xCR2032)
5,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	80 μ A 180 μ A	中端邮局秤, 厨房秤, 口袋秤	1,500 小时 (1xCR2032)
10,000	5 Hz	1 mV/V	1 kOhm 350 Ohm	300 μ A 700 μ A	中端口袋秤, 计数秤	2,000 小时 (1xCR2430)
80,000	5 Hz	2 mV/V	1 kOhm 350 Ohm	1.9 mA 4.5 mA	高端口袋秤, 计数秤	1,500小时 2 x AA

* 分度值为5个标准偏差的峰-峰值 (例如80000分度有效精度实际为400000内码)

2.8 时序

在 $V_{CC} = 3,3\text{ V} \pm 0,3\text{V}$, 外界温度为 $-40^{\circ}\text{C} \dots +85^{\circ}\text{C}$ 时否则另外说明。

表 2.10 振荡器时序

符号	32 kHz 参考振荡时钟	最小	典型	最大	单位
Clk10kHz	振荡启振时间	典型	10		kHz
ClkHS	高速参考振荡时钟	最大	4		MHz
toHSst	带陶瓷振荡器的振荡启振时间	单位	50	150	μ s

表 2.11 串行接口通信时序 (SPI)

符号	参数	最小	典型	最大	单位
fclk	串行时钟频率			1	MHz
tpwh	串行时钟, 脉冲宽度高平	500			ns
tpwl	串行时钟, 脉冲宽度低平	500			ns
tsussn	SSN 开启到 有效时钟沿	500			ns
tpwssn	在写循环之间的SSN 的脉冲宽度	500			ns
thssn	在SCLK下降沿之后的数据保持时间				
tsud	数据准备时间 在SCLK下降沿优先	30			ns
thd	在SCLK下降沿之前的数据保持时间	30			ns
tvd	在SCLK 上升沿后数据有效时间				ns

串行接口 (SPI兼容, 时钟相位=1, 时钟极性=0)

图 2.5 SPI - 写时序

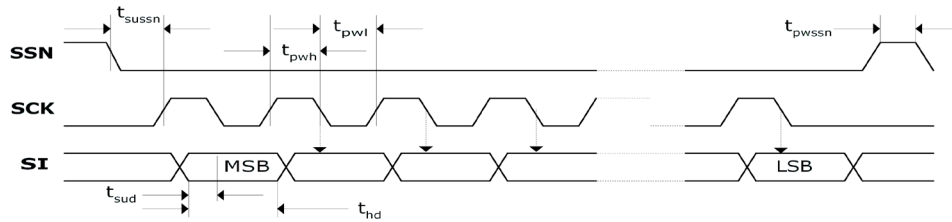
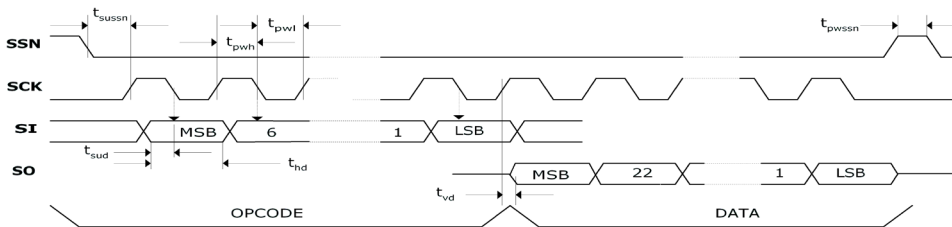


图 2.6 SPI - 读时序



2.9 管脚分配

PS081可以有裸片或者QFN56封装片。下面的图片和表格显示了管脚分配以及管脚的具体描述。

QFN56

图 2.7 管脚分配 QFN56

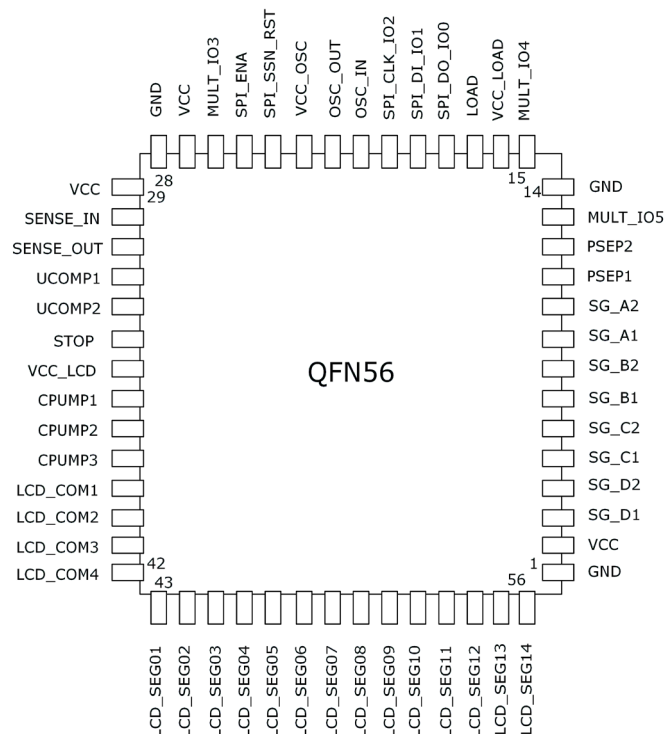


表 2.12 管脚描述 QFN56

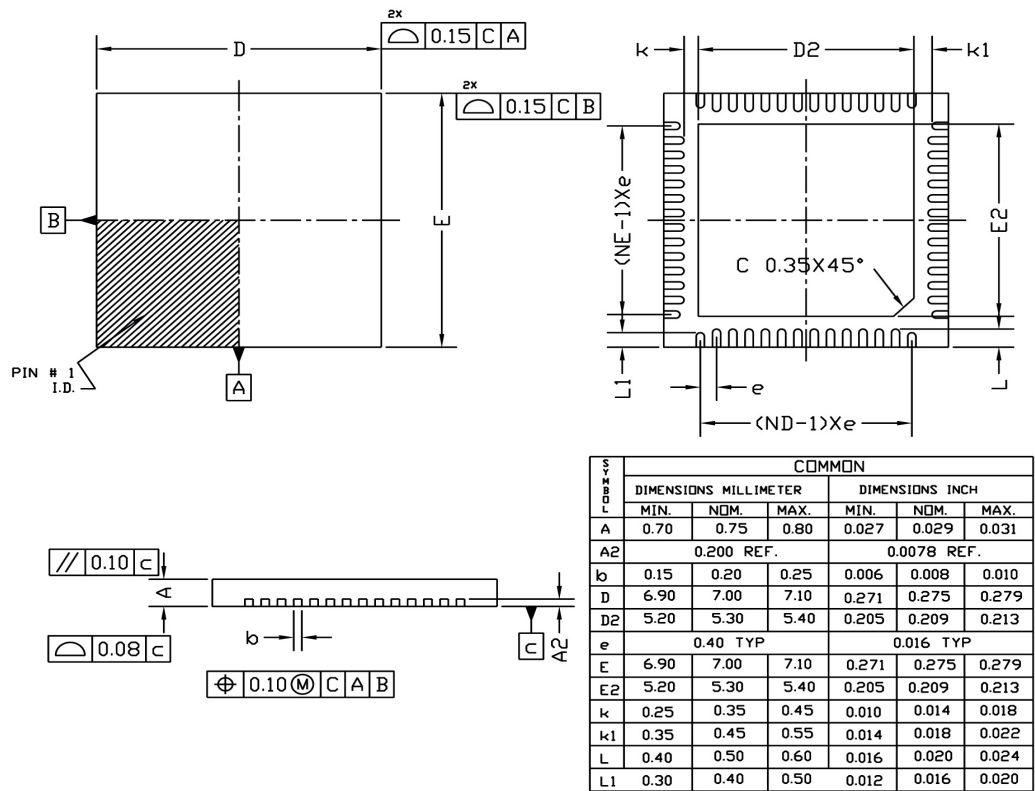
#QFN	名称	描述	类型
1	GND	地	
2	Vcc	数字部分供电电压, I/O, 4MHz-振荡	
3	SG_D1	端口1 半桥 D	N Open Drain
4	SG_D2	端口2 半桥 D	N Open Drain
5	SG_C1	端口1 半桥 C	N Open Drain
6	SG_C2	端口2 半桥 C	N Open Drain
7	SG_B1	端口1 半桥 B	N Open Drain
8	SG_B2	端口2 半桥 B	N Open Drain
9	SG_A1	端口1 半桥 A	N Open Drain
10	SG_A2	端口2 半桥 A	N Open Drain
11	PSEP1	端口1 温度测量	N Open Drain
12	PSEP2	端口2 温度测量	N Open Drain
13	MULT_I05	多功能 I/O5	Mult-I0
14	GND	地	
15	MULT_I04	多功能 I/O4	Mult-I0
16	Vcc_load	电源 load输出管脚	
17	Load	Load 输出接测量电容	P Open Drain
18	SPI_D0_I00	输出口 串行SPI 接口 或 I00	Mult-I0
19	SPI_DI_I01	输入口 串行SPI 接口 或 I01	Mult-I0
20	SPI_CLK_I02	串行 SPI 接口时钟 或 I02	Mult-I0
21	OSC_IN	4MHz 陶瓷晶振输入	
22	OSC_OUT	4MHz 陶瓷晶振输出	
23	VCC_OSC	4MHz 陶瓷晶供电电压	
24	SPI_CSN_RST SPI_SSN_RST	从机选择或者RST输入(高平有效)	Input with pull-down
25	SPI_ENA	串行 SPI接口使能	
26	MULT_I03	在比较器电路中选择惠私通选择器或者I03	Wheatstone select Mult-I0
27	Vcc	或者中断端口	
28	GND	数字部分供电电压 , I/O, 4MHz-振荡	
29	Vcc	数字部分供电电压, I/O, 4MHz-振荡	
30	SENSE_IN	内部CMOS比较器输入	Analog In
31	SENSE_OUT	内部CMOS比较器输出	Analog Out
32	UCOMP1	外部比较器电路连接	Analog Out
33	UCOMP2	外部比较器电路连接	Analog Out
34	STOP	测量信号stop输入	
35	VCC_LCD	LCD电源, 10kHz 晶振, 带隙	
36	CPUMP1	LCD 倍压和稳压	Analog Out
37	CPUMP2	LCD 倍压和稳压	Analog Out
38	CPUMP3	LCD 倍压和稳压	Analog Out
39	LCD_COM1	LCD 列驱动 1/2, 1/3, 1/4 duty	LCD Buffer
40	LCD_COM2	LCD 列驱动 1/2, 1/3, 1/4 duty	LCD Buffer

41	LCD_COM3	LCD 行驱动 1/3, 1/4 duty, 列驱动 1/2 duty	LCD Buffer
42	LCD_COM4	CD行驱动1/4 duty, 列驱动 1/2 , 1 /3 duty	LCD Buffer
43	LCD_SEG1	LCD 列驱动	LCD Buffer
44	LCD_SEG2	LCD列驱动	LCD Buffer
45	LCD_SEG3	LCD列驱动	LCD Buffer
46	LCD_SEG4	LCD列驱动	LCD Buffer
47	LCD_SEG5	LCD列驱动	LCD Buffer
48	LCD_SEG6	LCD列驱动	LCD Buffer
49	LCD_SEG7	LCD列驱动	LCD Buffer
50	LCD_SEG8	LCD列驱动	LCD Buffer
51	LCD_SEG9	LCD列驱动	LCD Buffer
52	LCD_SEG10	LCD列驱动	LCD Buffer
53	LCD_SEG11	LCD列驱动	LCD Buffer
54	LCD_SEG12	LCD列驱动	LCD Buffer
55	LCD_SEG13	LCD列驱动	LCD Buffer
56	LCD_SEG14	LCD列驱动	LCD Buffer

QFN56 封装信息

QFN56, 7x7mm², 0.4mm线宽

图 2.8 封装信息 (QFN56)

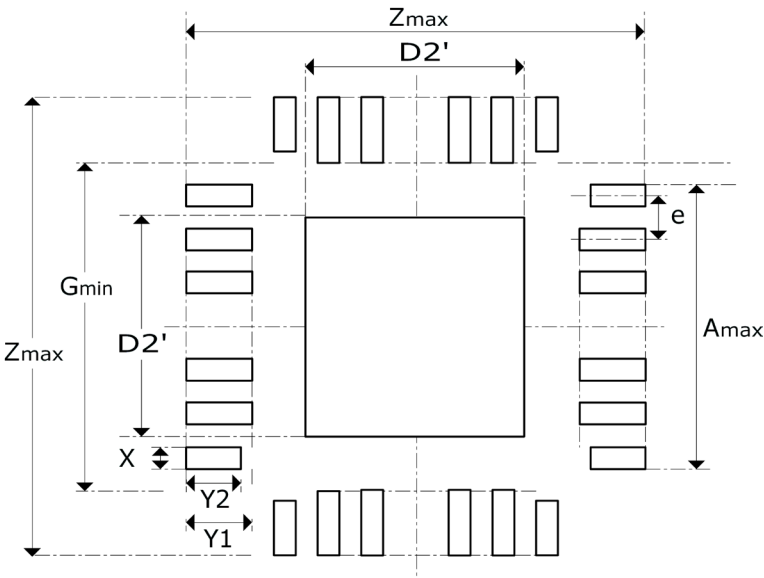


QFN56 推荐PAD管脚设置

图 2.9 Pad Layout

	mm	inch
e	= 0.4	0.016
G _{min}	= 6.3	0.248
Z _{max}	= 8.0	0.315
D2'	= 5.4	0.213
A _{max}	= 5.45	0.215
X	= 0.25	0.010
Y1	= 0.85	0.033
Y2	= 0.75	0.030

注:请不要减小地层的大小,不应包含任何过孔。



RoHS

PS081FN的QFN56封装满足RoHS

材料列表:	管脚边框	带有 PPF 边的C194 铜(NiPdAU)
	裸片盖胶	Ablebond 8600, ABlestik
	绑定线	金
	腌膜	CEL9220HF13H, Hitachi
	图标	激光

潮湿敏感度等级
芯片表面焊接

1 (JEDEC J-STD-020, 033)	
平均温度升高 (TL to Tp)	3 ° C/秒 最大
预热	
- 温度最小 (TSmin)	140 ° C
- 温度最高 (TSmax)	200 ° C
- 时间(最小到最大时间) ts	60 - 120 秒
相对温度的最大上升时间	
- 平均预热温度	3 ° C/秒 最高.
表面保持温度:	
- 温度 (TL)	220 ° C
- 时间 (tL)	30 秒
峰值温度 (Tp)	245 +0 -5 ° C
在峰值保持在 5° C 以内的最大时间	10 秒

Die

图 2.10 裸片管脚分配

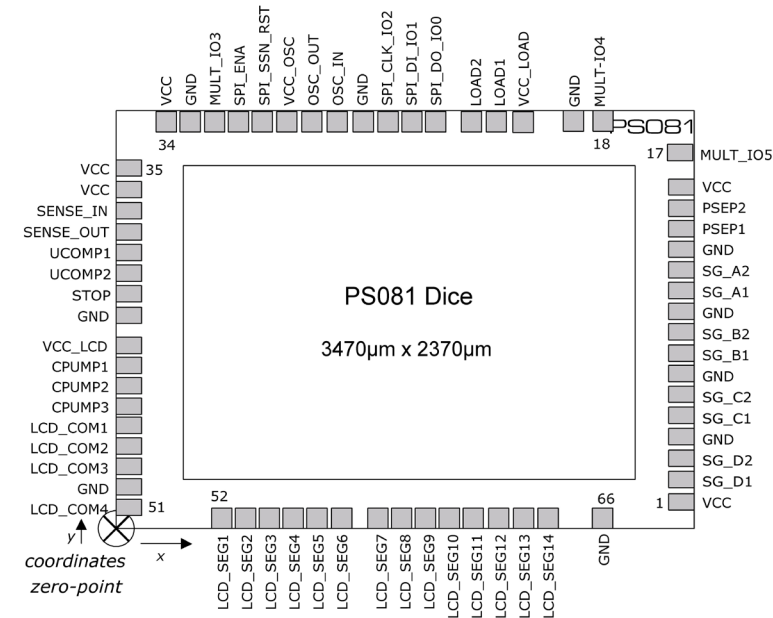


表 2.13 裸片管脚分配和位置

Pad编号	名称	描述	类型	X-坐标 中心点	Y-坐标 中心点
右面					
1	VCC	数字部分供电电压, I/O, 4MHz-振荡		3392	193.8
2	SG_D1	半桥D端口1	N Open Drain	3392	316
3	SG_D2	半桥D端口2	N Open Drain	3392	431
4	GND	地		3392	546
5	SG_C1	半桥C端口1	N Open Drain	3392	661
6	SG_C2	半桥C端口2	N Open Drain	3392	776
7	GND	地		3392	891
8	SG_B1	半桥B端口1	N Open Drain	3392	1006
9	SG_B2	半桥B端口2	N Open Drain	3392	1121
10	GND	地		3392	1236
11	SG_A1	半桥A端口1	N Open Drain	3392	1351
12	SG_A2	半桥A端口2	N Open Drain	3392	1466
13	GND	地		3392	1581
14	PSEP1	温度测量端口1	N Open Drain	3392	1696
15	PSEP2	温度测量端口2	N Open Drain	3392	1811
16	VCC	数字部分供电电压, I/O, 4MHz-振荡		3392	1926
17	MULT_IO5	多功能 I/O 端口 5	Mult-I/O	3392	2151
Top					
18	MULT_IO4	多功能 I/O 端口 4	Mult-I/O	2862	2286
19	GND	地		2730	2286
20	VCC_LOAD	电源 load输出管脚 1 和 2		2115	2286
21	LOAD1	到测量电容 Load的 输出	P Open Drain	1976.6	2286
22	LOAD2	到测量电容Load的 输出	P Open Drain	1835	2286
23	SPI_DO_IO0	输出串行SPI接口或者IO-0	Mult-I/O	1656.2	2286
24	SPI_DI_IO1	输入串行SPI接口或者 IO-1	Mult-I/O	1544.2	2286

25	SPI_CLK_I02	串行 SPI 接口或者I0-2时钟	Mult-I0	1432.2	2286
26	GND	地		1320.2	2286
27	OSC_IN	4MHz 陶瓷晶振输出		1208.2	2286
28	OSC_OUT	4MHz 陶瓷晶振输入		1096.2	2286
29	VCC_OSC	4MHz 陶瓷晶振供电电压		984.2	2286
30	SPI_CSN_RST SPI_SSN_RST	从机选择 或者RST 输入 (高平有效)	Input with pull-down	872.2	2286
31	SPI_ENA	串行SPI接口使能		760.2	2286
32	MULT_I03	在比较器电路选择惠私通选择器或者I03	Wheatstone select Mult-I0	648.2	2286
33	GND	地		536.2	2286
34	Vcc	数字部分电源电压, I/O, 4MHz-振荡		424.2	2286
Left					
35	Vcc	数字部分电源电压, I/O, 4MHz-振荡		83	2003.3
36	Vcc-SENSE	供电电压 SENSE 管脚		83	1891.3
37	SENSE_IN	内部CMOS比较器输入	Analog In	83	1779.3
38	SENSE_OUT	内部CMOS比较器输出	Analog Out	83	1667.3
39	UCOMP1	外部比较器电路连接	Analog Out	83	1555.3
40	UCOMP2	外部比较器电路连接	Analog Out	83	1443.3
41	STOP	测量信号stop输入		83	1331.3
42	GND	地		83	1219.3
43	VCC_LCD	LCD电源, 10kHz 振荡., 带隙		83	1045
44	CPUMP1	LCD 倍压和稳压	Analog Out	83	933
45	CPUMP2	LCD倍压和稳压	Analog Out	83	821
46	CPUMP3	LCD倍压和稳压	Analog Out	83	709
47	LCD_COM1	LCD 列驱动 1/2, 1/3, 1/4 duty	LCD Buffer	83	597
48	LCD_COM2	LCD 列驱动1/2, 1/3, 1/4 duty	LCD Buffer	83	485
49	LCD_COM3	LCD 行驱动 1/3, 1/4 duty, 列驱动 1/2 duty	LCD Buffer	83	373
50	GND	地		83	261
51	LCD_COM4	LCD行驱动1/4 duty, 列驱动 1/2 , 1/3 duty	LCD Buffer	83	149
Bottom					
52	LCD_SEG1	LCD 列驱动	LCD Buffer	612.6	83
53	LCD_SEG2	LCD 列驱动	LCD Buffer	724.6	83
54	LCD_SEG3	LCD 列驱动	LCD Buffer	836.6	83
55	LCD_SEG4	LCD 列驱动	LCD Buffer	948.6	83
56	LCD_SEG5	LCD 列驱动	LCD Buffer	1060.6	83
57	LCD_SEG6	LCD 列驱动	LCD Buffer	1172.6	83
58	LCD_SEG7	LCD 列驱动	LCD Buffer	1347	83
59	LCD_SEG8	LCD 列驱动	LCD Buffer	1459	83
60	LCD_SEG9	LCD 列驱动	LCD Buffer	1571	83
61	LCD_SEG10	LCD 列驱动	LCD Buffer	1683	83
62	LCD_SEG11	LCD 列驱动	LCD Buffer	1795	83
63	LCD_SEG12	LCD 列驱动	LCD Buffer	1907	83
64	LCD_SEG13	LCD 列驱动	LCD Buffer	2019	83
65	LCD_SEG14	LCD 列驱动	LCD Buffer	2131	83
66	GND	地		2928	83

裸片尺寸和Pad and Pad开窗尺寸

裸片的绝对尺寸为2.37 x 3.47 mm，晶圆厚度为 725 μ m。IC主要应用为片上系统单芯片应用 (COB)。因此晶圆绑定的开窗尺寸需要配合绑定的机器：

宽： 90 μ m

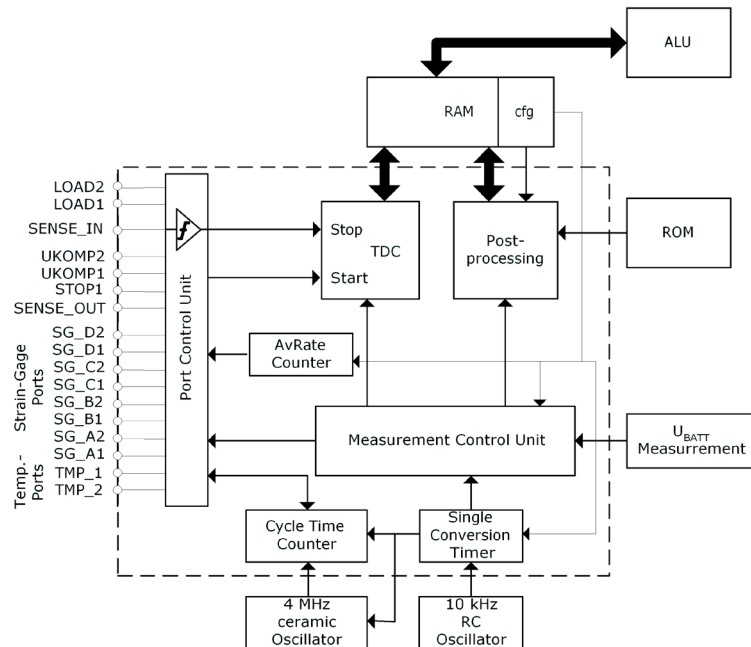
高： 116 μ m

目录	页面
3 高端转换器	3-2
3.1 概述.....	3-2
3.2 测量原理.....	3-2
3.3 连接金属应变传感器	3-3
3.3.1 半桥	3-3
3.3.2 全桥	3-4
3.3.3 并联全桥 (优化零点偏移)	3-5
3.3.4 全桥连接为半桥 (低功耗测量连接方式).....	3-5
3.3.5 惠斯通桥	3-6
3.3.6 四驱桥模式 (4 个半桥传感器).....	3-7
3.3.7 两个分别的半桥	3-8
3.3.8 6-线制测量技术	3-8
3.4 放电电容、周期时间、求平均.....	3-10
3.4.1 放电电容(Clload)	3-10
3.4.2 周期时间 (cytime)	3-11
3.4.3 Stretched拉伸模式的周期时间.....	3-11
3.4.4 平均率 (avrate).....	3-12
3.4.5 通过平均的方式提高测量精度	3-12
3.4.6 测量分辨率和转换精度.....	3-14
3.5 模式和时序	3-15
3.5.1 连续模式	3-15
3.5.2 单一转换模式	3-15
3.5.3 拉伸模式	3-16
3.5.4 模式选择标准	3-19
3.5.5 连续模式中的转换时间/测量刷新率.....	3-20
3.5.6 转换时间/测量频率 (单一转换模式)	3-20
3.5.7 比较器	3-21
3.5.8 温度测量	3-23
3.6 预处理.....	3-25
3.6.1 四角秤的角差补偿.....	3-26
3.6.2 传感器的Gain 和 Offset 漂移的温度补偿	3-26
3.6.3 Rspan的一些注释.....	3-28
3.6.4 温度增益漂移的非线性度.....	3-29
3.6.5 PS081自身的Gain漂移—通过MULT_PP进行优化.....	3-31
3.6.6 PS081本身的零点漂移	3-33
3.6.7 MuLT_UB - 电源偏移抑制	3-34

3 高端转换器

3.1 概述

图 3.1 概述



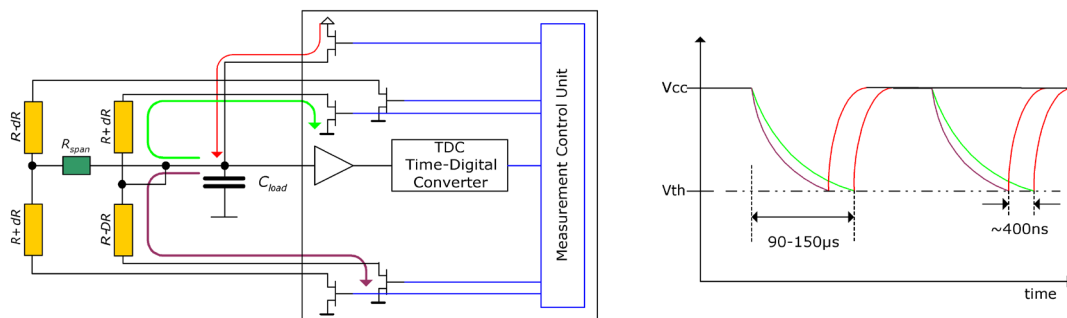
基于PICO STRAIN测量原理的转换器带有应变电阻测量端口(SG_Ax 到 SG_Dx)可以测量:

- 4个独立的半桥 (4角模式)
- 2个半桥组成的全桥
- 2个独立的半桥
- 1个经典的惠斯通桥
- 1个单独得半桥

3.2 测量原理

应变得测量本身是通过测量放电时间来间接体现的。放电时间是测量应变电阻通过一个放电电容 C_{load} 放电来获得。正相变化和反向变化的应变电阻的放电时间都会被进行测量。那么两个放电时间的比值则会反映应变电阻的变化信息。时间测量是通过高精度内部时间单元完成, 15ps的测量精度。(通过平均可以达到0.5 ps)。

图 3.2 测量原理



在这个章节当中将会解释实现高端测量功能需要的元器件，以及需要设置的参数和外部元器件的尺寸选择。

有很多种方法可以将应变片连接到PS081，在这个章节中我们将着重介绍一下。

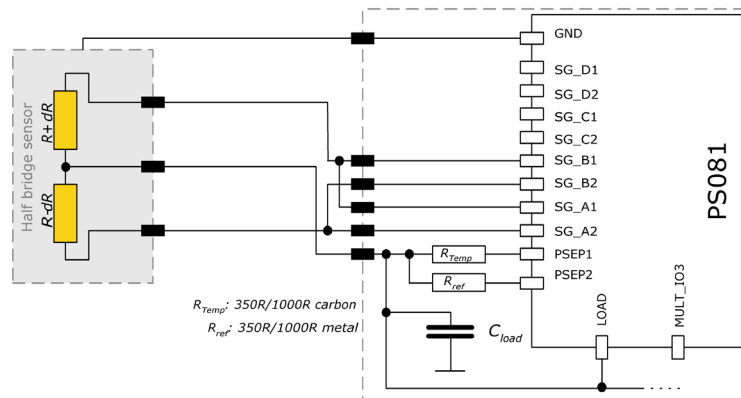
3.3 连接金属应变传感器

注意:

为了获得好的测量结果，必须要将传感器的壳体与电路板的地相联接，一个简单的标准导线就满足要求了。

3.3.1 半桥

图 3.3 半桥



注意:

为了得到更好的温度零点漂移特性，半桥连接需要如全桥连接方式一样进行连接。这需要把bridge设置为1[寄存器3, bridge (1:0) =1]

乘法因数必须为正，例如：Mult_Hb1=+1, Mult_Hb2=+1。

在这样的连接情况下电路的零点偏移将会同全桥方式一样完全满足OIML规定的标准。

在这样的连接情况下电路的零点偏移将会同全桥方式一样完全满足OIML规定的标准。

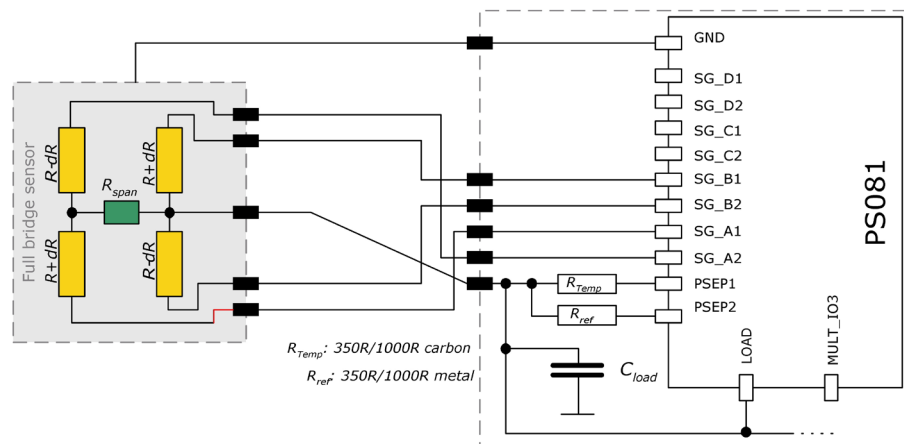
为了获得最快的测量速度，则应该以半桥的连接方式而不是全桥的连接方式连接半桥[寄存器3, bridge (1:0) =1, 只连接SG_A1和SG_A2两个测量端口]。

当AVRate=2时，测量速度最大可能达到1KHz。那么一个系统的零点偏移将会加在测量当中，几乎在所有电路中都会有这样的零点偏移。这个偏移大概是 $\pm 6\text{nV/V}$ 。这个零点偏移主要是由于在SG_A2上的绑定长度比SG_A1端口的绑定长度要长。而绑定线的电阻没有被芯片的R_{DS(on)}补偿去掉。因此，这个系统得偏移可以通过在PCB布线中将SG_A1的走线加长来获得。

当应用一个标准的PCB带有35um厚铜层的时候，从芯片到连接传感器端口SG_A1的长度应该比SG_A2（应用8mm宽的走线）长（待定）毫米。这个补偿的方法是非常有效和可靠的，通过补偿后的零点完全满足OIML的要求。

3.3.2 全桥

图 3.4 全桥连接方式

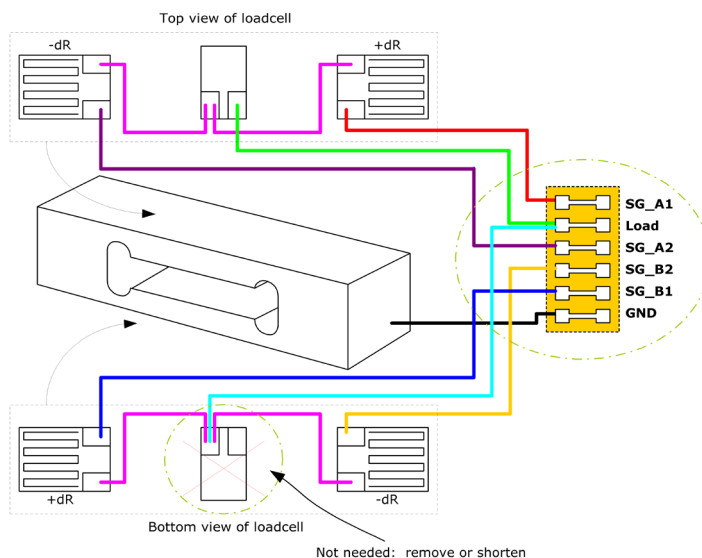


注意：

上图的连接方式是标准的PICO STRAIN电桥连接方式（由两个半桥加一个可以选择的补偿电阻（Rspan）组成的全桥）。bridge设置为1[寄存器3, bridge(1:0)=1] 乘法因数必须为正，例如：Mult_Hb1=+1, Mult_Hb2=+1。因此必须要严格遵循传感器正向应变和负向应变的连接方向。

惠斯通电桥方式连接的传感器只需要按照下图所示在接线面板中来改变应变片的接线就可以调整到我们PICO STRAIN连接方式。

图 3.5 调整过的应变电阻接线方式



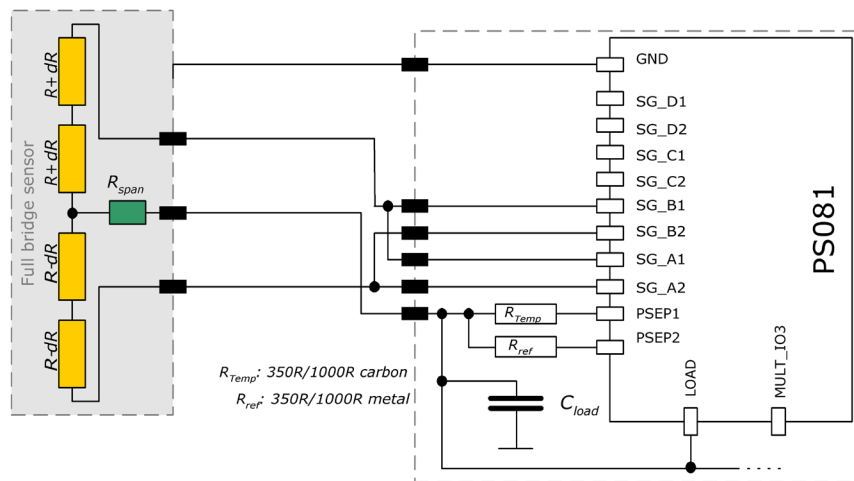
与惠斯通桥连接相比，PICO STRAIN的全桥连接在测量精度上要高0.6位（相当于惠斯通桥的1.5倍或者更高些）。

3.3.3 并联全桥 (优化零点偏移)

这个模式不再推荐使用（2009年12月被删除掉），详细信息请参见第7.1章节。

3.3.4 全桥连接为半桥 (低功耗测量连接方式)

图 3.6 低功耗全桥连接方式



注意：

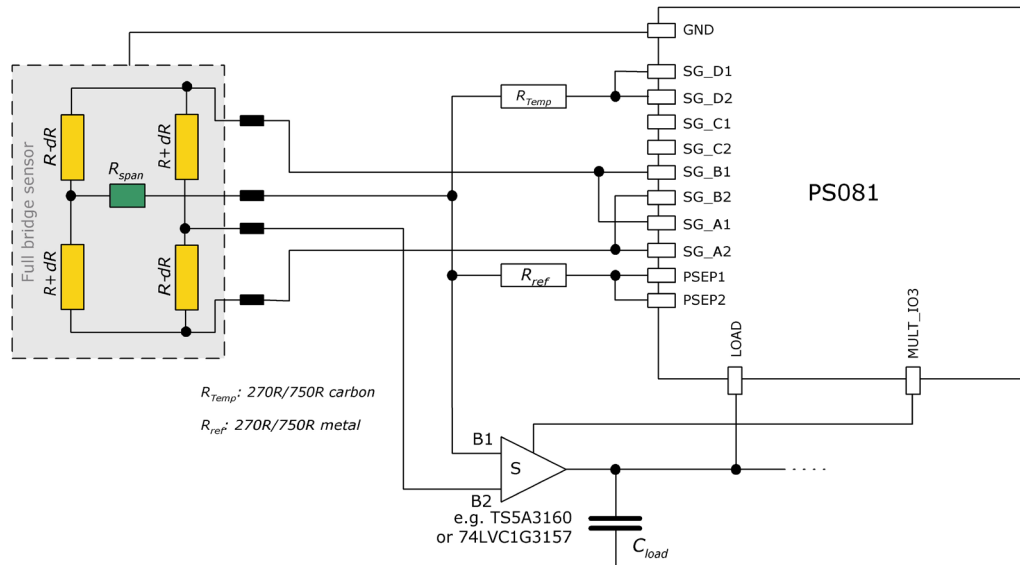
如果所有的应变片都是装在传感器的同一侧（顶部或者底部），则可以采用这种接线。这种连接将两个应变电阻串联起来形成一个阻值为2k欧姆的半桥。这种连接方式，可以把传感器中的电流减小一半，因此最适合于低功耗的应用，例如：太阳能驱动的应用。

Bridge设置为1[寄存器3, Bridge (1:0) =1]

乘法因数必须为正信号, 例如: Mult_Hb1=+1, Mult_Hb2=+1。

3.3.5 惠斯通桥

图 3.7 惠斯通桥连接方式



对于惠斯通桥的连接方式，需要一个外部的模拟开关来进行控制。我们推荐的型号为TS5A3160, 因为这个元件在供电电压低于2.7V的情况下有一个非常好的特性。在供电电压高于3V的情况下74LVC1G3157同样也是一个不错的选择。

注意：

在惠斯通模式下系统精度将会下降0.6位。因此我们仅在以下两种情况下推荐使用惠斯通电桥连接：

1. 传感器线长度大于1m
2. 你不想改变传感器接线时的初次测试

Bridge设置为1[寄存器3, Bridge (1:0) =1]

乘法因数的符号必须是相反的, 例如Mult Hb1=+1, Mult Hb2=-1。

如果惠斯通桥带有一个Rspan电阻，那么在寄存器TKGain的标准设置应该0.75[ConfigReg08]。系数0.75不会改变测传感器的span补偿性能。任何情况下，”Mod_Rspan”都必须设置为1[ConfigReg01, bit6]。

为了避免导线的反射，在惠斯通模式中我们推荐应用磁珠。磁珠请连接到PS081的两条测量线上。推荐应用的磁珠种类为普通的（SMD类型）的铁氧体，在100Mhz的阻尼为1000ohm，DC直流低阻抗（<0.1欧姆）。那么在这个情况下将会获得更低的零点偏移以及更好的EMI特性。

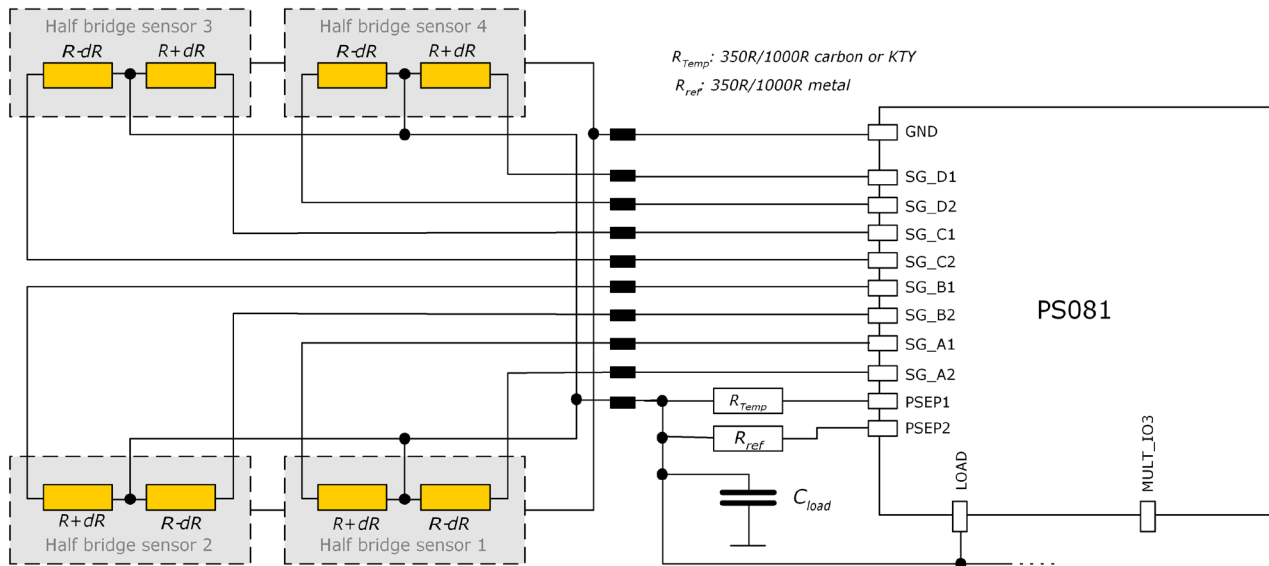
警告:

只能使用带一个Rspan或不带Rspan（非补偿）的惠斯通桥。

如果使用带两个Rspan的惠斯通桥，PICO STRAIN原理的芯片将不能正常工作。。

3.3.6 四驱桥模式 (4 个半桥传感器)

图 3.8 四驱桥连接方式



在有些情况下需要用到4个传感器。那么每一个半桥都会被连接到一个端口，这是一个典型的连接。例如：四驱人体秤。这种连接既可以读出每一个半桥的测量结果也可以读出总的测量结果。

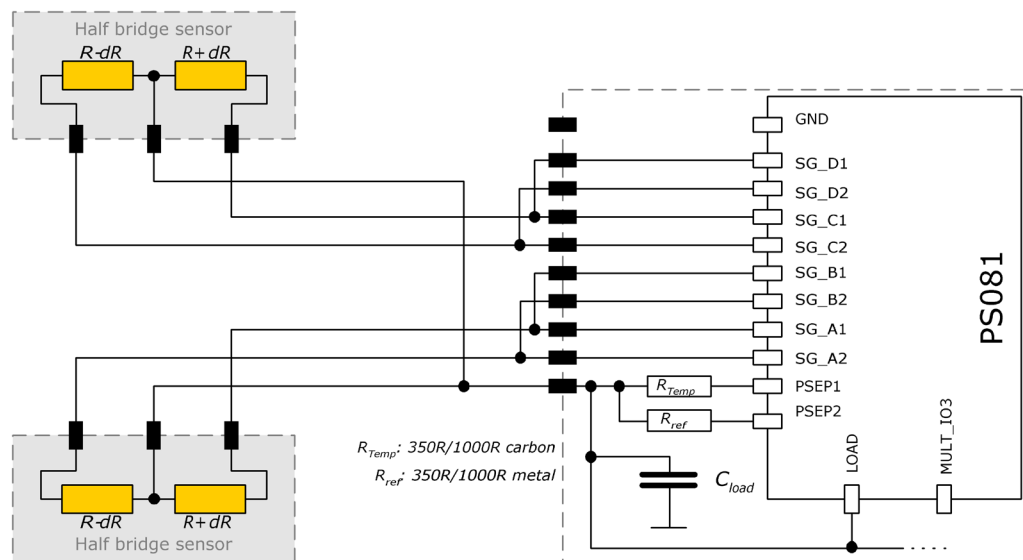
Bridge设置为3[寄存器3, Bridge (1:0) =3]

每一个半桥都有一个自己的乘法系数。这将允许对每一个半桥分别进行增益的软件补偿。

所有半桥的乘法因数必须为正，例如Mult_Hb1=+1, Mult_Hb2=+1, Mult_Hb3=+1, Mult_Hb4=+1。

3.3.7 两个分别的半桥

图 3.9 两个单独半桥的连接方式



注意：

一般来说两个半桥将组成一个全桥（输出一个结果）。然而，有时候也需要知道单个半桥的测量结果，这时候两个半桥就需要必须分开测量。在这种情况下，这两个半桥的连接和上图所示的四驱连接模式中相同，可以分别读出每个半桥的结果。这种连接方式可以保证结果是经过增益补偿的。

每个半桥的结果都可以按照下面的公式来计算：

$$HB1 = (A - B) / 2, \quad HB2 = (C - D) / 2$$

如果您想要分别读出两个半桥的结果并且需要很低的offset偏移，请按照全桥连接法建议的方法连接，并且请联系acam团队获取更进一步的信息。

bridge设置为3[寄存器3，桥（1:0）=3]

所有的乘法因数必须为正，例如Mult_Hb1=+1, Mult_Hb2=+1, Mult_Hb3=+1, Mult_Hb4=+1。

3.3.8 6-线制测量技术

PS081可以应用在6线制的传感器。图3.10和3.11显示了6线制技术主要是应用于补偿导线电阻所带来的影响。连接PS081导线的电阻将会被作为比较器增益误差补偿的一部分被补偿（MULT_PP，请参见3.6.2章节）。

图 3.10 6-线制的PICOstrain连接方式

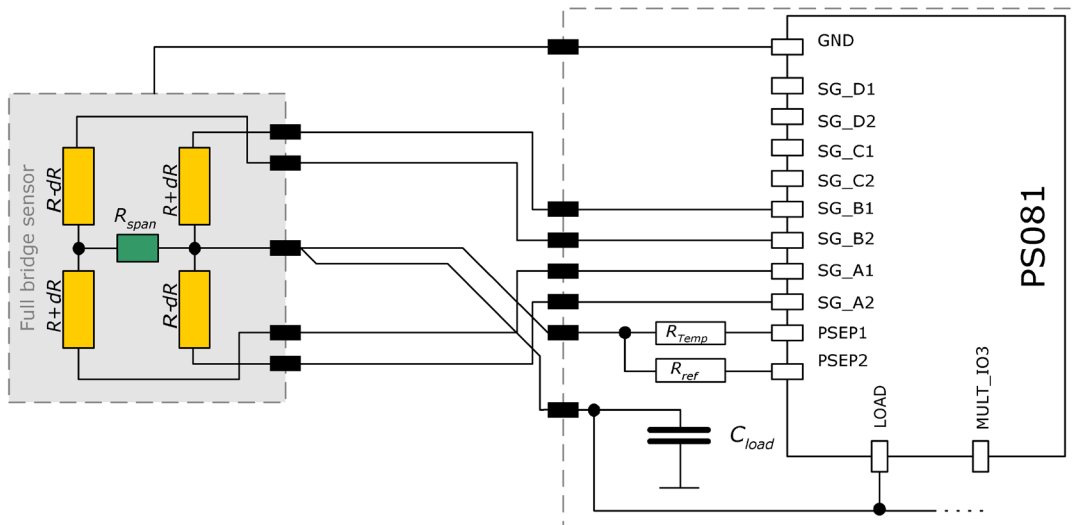
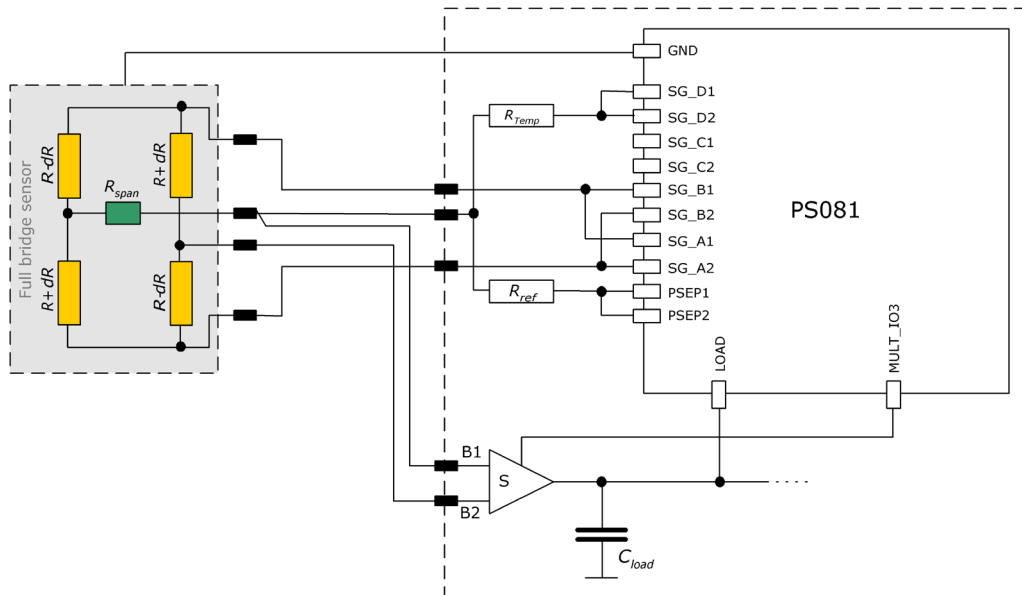


图 3.11 6-线制的惠斯通连接方式



3.4 放电电容、周期时间、求平均

3.4.1 放电电容(Cload)

放电电容是电路中的一个重要部分，它直接影响到测量的精度和温度稳定性。因此，我们推荐使用以下电容值和材料的电容。

Rsg = 350R → Cload = 300nF to 400nF

Rsg = 1000R → Cload = 100nF to 150nF

Cload可通过公式 $t = 0.7 \cdot R_{sg} \cdot C_{load} \approx 100\mu s - 150\mu s$ 计算得出。

推荐的材料：

- COG* (用于高精度测量)
- CFCAP** (也不错，但比COG稍差)
- X7R (在温度稳定性上有一些损失)
- Polyester (在温度稳定性上有一些损失)

我们不推荐使用ZOG电容器！

* 村田公司的GRM31 系列电容就是COG电容，单个电容值最高可达100nF

** CFCAP材料的电容可使用日本太阳诱电公司的多层陶瓷电容

注意:

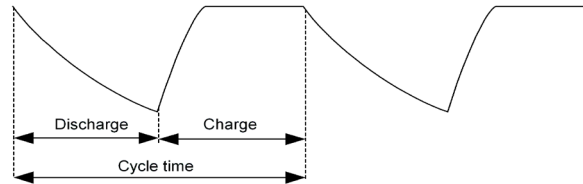
COG电容在高端应用中绝对是第一选择（比如6000分度OIML或者更高标准的衡器）。CFCAP对于高端衡器和OIML衡器来说也是不错的选择。X7R 由于价格比较低因此是家用秤的第一选择。但是这个材料的电容在低温（< +5摄氏度）时会带来额外的增益漂移。

对于消费类的低端衡器来说很多其他材料的电容也都适合（例如：Polyester）。

3.4.2 周期时间 (cycetime)

周期时间是指连续两次放电时间测量之间的时间间隔。它包含了一次放电时间和对Cload再次充电的时间。下图说明了充放电之间的关系。

图 3.12 周期时间



放电时间是由应变电阻的阻值和所选的电容Cload的电容值所决定的。推荐的放电时间是80us~150us (3.3V情况下)。充电时间必须足够长以保证Cload被充满，充电时间一般为整个周期时间的30%。如果周期时间设置得太小（在放电时间的范围内或者更小）就会出现溢出。

周期时间在寄存器2, cycetime(13:4)中进行设置。

周期时间通常由高速时钟产生，可以以2us的单位来进行设置。唯一的例外是在“stretched拉伸模式”中（参考“拉伸模式”章节）。此模式下，周期时间是由内部10KHz晶振产生的，因此它将以100us为单位进行设置。

例如：

`cycetime(13:4)=80` → $80 \times 2\mu\text{s} = 160\mu\text{s}$ 用于除了stretched拉伸模式外的所有模式

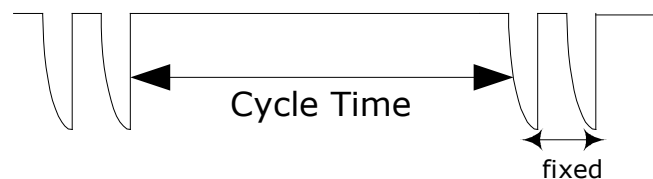
`cycetime(13:4)=10` → $10 \times 100\mu\text{s} = 1\text{ms}$ 仅用于stretched拉伸模式

最小周期设置推荐为放电时间的1.4倍，例如：放电时间为100us则最小周期设为140us。

3.4.3 Stretched拉伸模式的周期时间

在stretched模式中参数cycetime具有一个特殊的作用。它不再是放电时间+充电时间，是以100us的倍数来定义两个放电周期测量之间的时间间隔。

图 3.13: stretched拉伸模式的周期时间



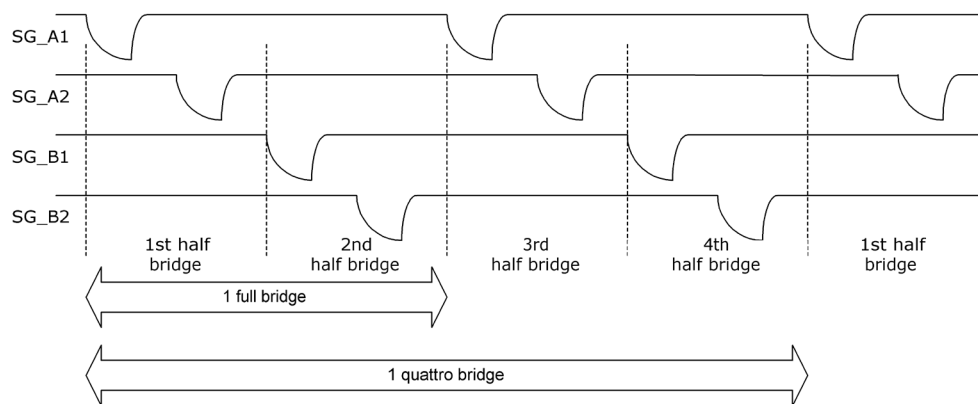
在stretched拉伸模式中有一些参数需要调整，请参阅接本章后面的3.5.3章节的“Stretched 模式的设置”。

3.4.4 平均率 (avrate)

传感器中半桥的应变片数量决定了需要进行几次放电循环才能完成一次完整的比率测量：

半桥	→	2 次循环
全桥	→	4 次循环
四驱模式	→	8 次循环

图 3.14 一次完整测量的放电次数

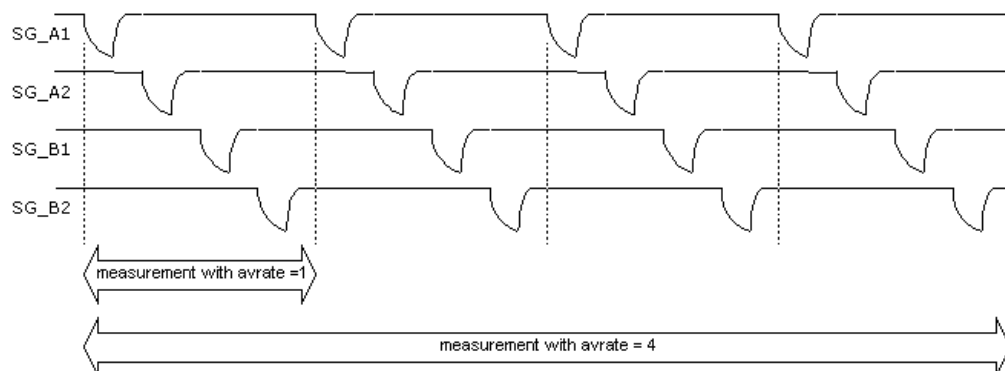


在每种模式中把一次测量所需的全部循环合在一起算作一次采样（AVRate=1）。这也是一次完整的比率测量所需的最小采样数。

3.4.5 通过平均的方式提高测量精度

PS081中，通过内部求平均的方法可以提供测量精度。可以在register2的参数AVRate中设置求多少次采用的平均值。通过取平均值，测量结果的标准偏差会提供所求平均数的均方根次倍。下图以全桥为例来做说明：

图 3.15 平均率



在这个例子中一个序列是由4次采样组成，每一次采样由4次放电循环组成，所以总的来说一次测量由16次放电循环组成。

在采样中除了放电循环外，还有一些其他的测量，如增益补偿或为了增强稳定性的而进行的伪测量。所有这些测量一起才算一次真正的测量。换一种说法就是一个测量序列包含所有得到最终结果需要的测量。它也规定了总的转换时间。更多有关转换时间方面的细节，请参见3.5.5的章节“转换时间”和3.5章节“模式”。

当然，取平均的采用次数决定刷新率。当精度提高 $1/\sqrt{\text{avrate}}$ 时，最大刷新输出率就会减小 avrate 倍。

采样次数 (avrate) ↗	→	精度 ↗
	→	最大输出率 ↘

最小功耗也会受到采样次数 (avrate) 的影响：

采样次数 (avrate) ↗	→	最小电流 ↗
-----------------	---	--------

建议：

我们强烈的建议不要使用AVRate=1，原则上AVRate=1时也能工作，但是漂移会显著增加，只能用在精度要求比较低的领域。建议最小AVRate=2。同时建议在avrate小于50时，不要使用奇数，例如：不要用AVRate=7，用AVRate=6或者AVRate=8替代

重要：在地测量刷新率情况下 (≤ 32) 系数PS081adjust(寄存器3, Bit (9: 4)) 应该设置为2倍的avrate。例如avrate为8，那么设置Ps081adjust为16。

3.4.6 测量分辨率和转换精度

在这个文档中我们经常会提到测量的分辨率和转换精度，这两个概念是不同的：

- **分辨率：**对于数字电路来说是这颗芯片可以给出显示的最小值。HB0结果寄存器的基本结构是24位，那么MSB最高位代表的为符号显示（2进制补码形式）。以10进制的数字表示那么测量结果的显示范围为-8388608（0x800000）到 +8388607（0x7FFFFFFF）。那么一个LSB则代表10nV/V（2mV/V除以200000）。例如：在3V电压情况下，1个LSB为10nV/V×3V=30nV.
- **转换精度（有时候也会被称为”准确度”）：**这是转换器所能给出的最大测量准确度，一般是通过测量的标准偏差或者RMS（均方差）噪声给出。测量数字的精度一般是由有效精度位（ENOB）定义。在PS081测量中，测量的RMS噪声在3.3V的时候最低可致10nV，或者换一种表达，ENOB达到19.5位（相对应2mV/V）。一个对于在不同输出频率下的转换精度在章节2.4的表格中有给出。另外一个大致的估算可以通过在下面给出的公式来推算。

在AVRATE =1（仅为了计算，不推荐实际使用）、一个半桥、在快速模式下推荐的放电时间为90~150 μs以及2mV/V 激励时，PS081的基本转换精度为：

带有内部比较器：13.3 有效位

带有外部双极性比较器：13.8有效位

在AVRATE值更高时精度的计算如下：

$$ENOB = ENOB[AVRate=1] + \frac{\ln(\sqrt{AVRate * Bridgefactor})}{\ln(2)}$$

桥臂-系数： 全桥 2
 四驱 4

例1:

AVRate = 12, 四驱电桥, 内部比较器

ENOB = 13.3 + $\ln(\sqrt{(12*4)})/\ln(2)$ = 13.3 + 2,8 = 16.1 有效位 = 70.000 有效分度 = 1万峰峰值分度（快速模式，无SINC-滑动平均滤波）。

例 2:

avrate = 450, 全桥, 外部比较器

ENOB = 13.8 + $\ln(\sqrt{(450*2)})/\ln(2)$ = 13.8 + 4.9 = 18.7 有效位 = 425,000 有效分度= 7万峰峰值显示分度（快速模式不带SINC-滤波）。

例3:

avrate（平均次数） = 100, 全桥, 外部比较器, 以 nV RMS的单位表示

ENOB = 13.8 + $\ln(\sqrt{(100*2)})/\ln(2)$ = 13.8 + 3.8 = 17.6 有效精度位

$2^{17.6}$ = 198,700 有效分度 在2mV/V灵敏度 3V供电情况下

3V x 2mV/V = 6000μV 除以198,700 = 30.2nV RMS

有效精度位（ENOB）在这个公式当中是相对应于2mV/V的传感器灵敏度的。如果你的传感器灵敏度不同的话，那么需要根据你的灵敏度来进行相应计算。比如：我们在公式中计算出测量的有效精度位为18.7位，而你应用的传感器仅为1mV/V, 那么这相当于有一个2倍的损失，那么也就是1位，在这种情况下下的ENOB为17.7位。

有效精度位（ENOB）在当应用惠斯通电桥而不是我们的PICO STRAIN全桥连接情况下也有一定损失。原因是由于惠斯通桥的测量情况下，应变的灵敏度会自动降低1/3，主要是在惠斯通桥测量的时候，当每测量一个应变片时，另外3个应变片是与这个被测应变片并联的。因此ENOB会有-0.6位的损失。

3.5 模式和时序

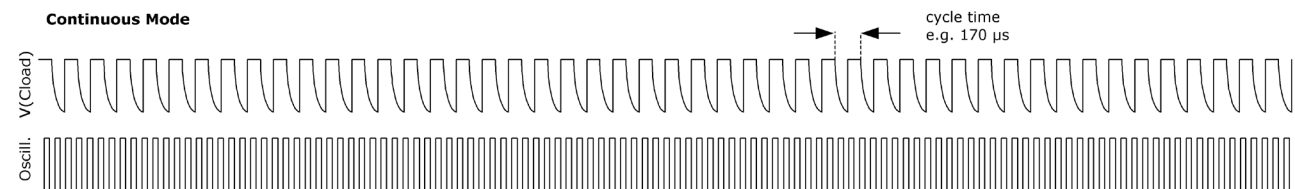
PS081有3种基本的操作模式，也可以把这三种模式组合起来使用。这3种模式与采样频率和4MHz晶振的工作时间有关。因此，选择何种模式会影响到测量结果的稳定性和功耗。

基本模式有：

- 连续模式
- 单一转换模式
- 拉伸模式

3.5.1 连续模式

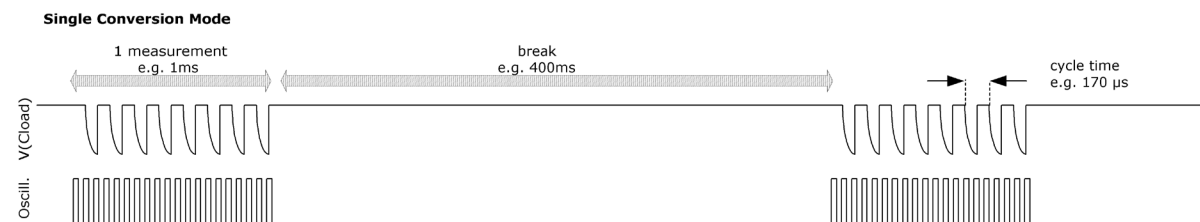
图 3.16 1、连续模式



芯片不间断地测量放电的时间，晶振一直开启。这种模式适用于高精度应用。在允许电流消耗>500uA的应用中，此模式是的标准应用模式。

3.5.2 单一转换模式

图 3.17 单一转换模式



芯片完成一次完整的测量之后进入睡眠模式。晶振只是在测量时才开启。这种模式功耗最

低，是人体秤的最佳选择。

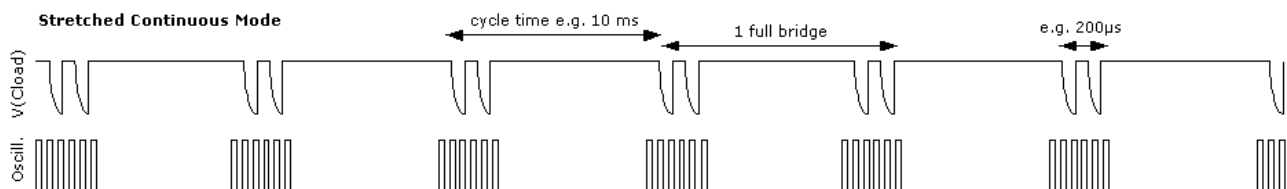
纯单一转换模式必然会欠采样，因此只能应用在机械性稳定的系统中，如人体秤。欠采样的结果就是称重传感器的机械振动将会导致不稳定的测量数据。

3.5.3 拉伸模式

拉伸模式既具有测量次数少、功耗低的优势又合理地分配了测量，避免了欠采样状况的出现。在这种模式下，总循环次数并不改变，只是拉长了分布周期，放电周期也因此被拉长。

3.5.3.1 拉伸连续模式

图 3.18 拉伸连续模式

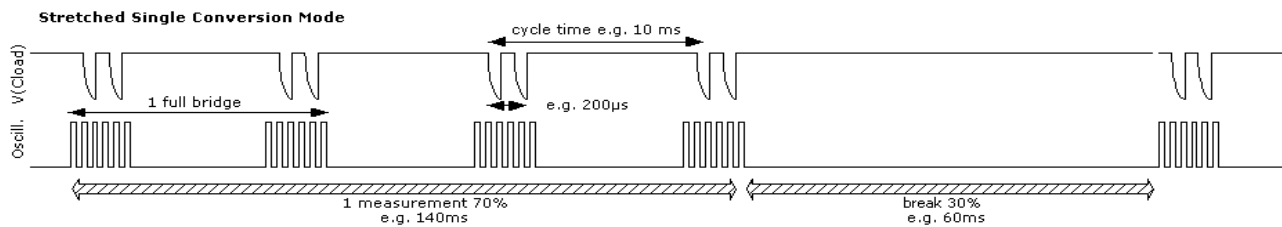


拉伸连续模式把拉伸模式和连续续模式结合在一起。半桥的两次放电时间测量之间的间隔变长了。晶振只有在进行每个半桥测量的时候开启。

这种模式适用于高精度、低功耗（电流 $<500\mu A$ ）的应用。对输入信号也具有好的频域响应（如传感器的机械振动）。此响应可用奈奎斯特采样定理来计算。在拉伸连续模式下搭配一个好的防振动滤波软件，就能得到低功耗低振动的最佳状况。如果做太阳能厨房秤，则推荐使用这种模式。

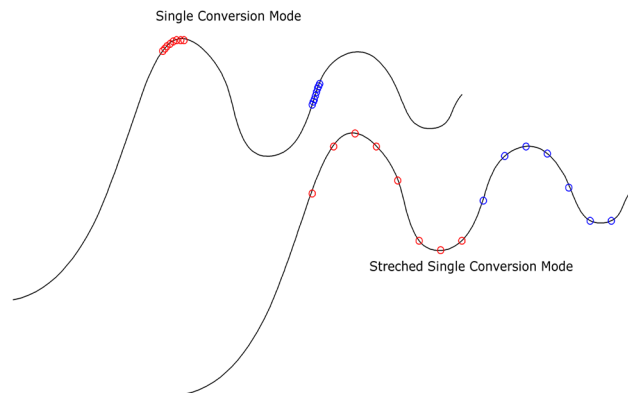
3.5.3.2 拉伸单一转换模式

图 3.19 Stretched拉伸单一转换模式



PS081为机械振动敏感的衡器（如厨房秤）提供了拉伸模式与单一转换模式相结合的模式。这种模式中，一个半桥的两个电阻的测量是紧跟在一起的，但是接下来的一对放电时间的测量将会进行延迟。因此，这样的连续的采样是可以覆盖机械振动整个周期区域范围的。由于这种采样形式，只要每两次测量的间隔时间小于整个时间的30%，如上图，结果通常就是稳定的。

图 3.20 欠采样



另外，晶振仅在测量放电时间的时候开启。但是，因为晶振需要一段时间才能达到满幅值，晶振总的工作时间比在纯单一转换模式中要长。因此，拉伸单一转换模式的功耗要比纯单一转换模式的功耗稍大一点。

4个主要的参数定义了操作模式

single_conversion:	选择是连续的测量还是单一分离测量。
stretch:	选择4M晶振是持续工作还是只有在测量放电时间的1到2个放电周期内工作。（推荐2个放电周期）。
cytime:	决定单次或两次充/放电时间的间隔。使用4MHz的时钟，在拉伸模式下使用10MHz时钟。
avrate:	采样的平均次数。定义一次完整的测量所包含的测量次数（通过内部平均）。

single_conversion / continuous

配置: 寄存器 2, Bit2: single_conversion。

single_conversion = 0: 选择连续模式, 在这个模式中PS081持续测量。4MHz的晶振持续工作, 其功耗大概为130uA @ 3.0V

single_conversion = 1: 选择单一转换模式, 在这个模式中PS081完成一次完整的测量后在进行单次转换计时期期间关闭4MHz晶振。

stretch

配置: 寄存器 3, Bits 12, 13: stretch

stretch = 0: 关闭

stretch = 1: 4MHz晶振仅在测量单次放电时间期间起振。周期时间（两次相邻放电测量之间的时间间隔）的计算以10kHz晶振为基准。

stretch = 2 or 3: 4MHz的晶振只有在测量一个单一半桥测量期间起振（2次放电时间测量）。周期时间（两次相邻半桥时间测量之间的时间间隔）的计算以为10kHz晶振为基准。一个半桥两次放电时间测量之间的间隔: 在Stretch =2时为200us; 在Stretch=3时为300us。

拉伸模式设置

在拉伸模式下, 有几个参数需要配置, 它们是:

stretch[13:12], Configreg_03

cyctime[13:4], Configreg_02

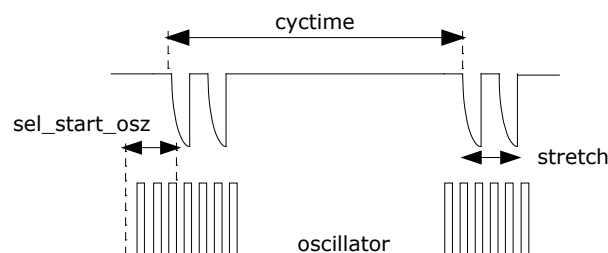
sel_start_osz[19:17], Configreg_03

single_conversion [2], Configreg_02

tdc_conv_cnt[23:16], Configreg_00

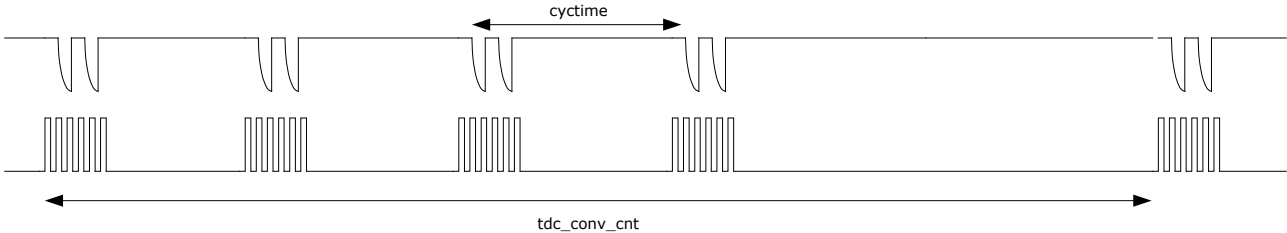
有了这些参数可以调整拉伸模式, 接下的两张图片说明了这些参数是如何应用的, 一张图是连续拉伸模式另一张是单一转换拉伸模式。

图 3. 21 在连续拉伸模式下的参数



参数cyctime定义了下两次放电测量之间的间隔。通过参数Stretch，就可以确定两次放电周期之间的时间间隔。参数sel_start_osz规定了在测量放电周期之前晶振的启动时间。

图 3.22 拉伸单一转换



在单一转换延伸模式中，参数的设置基本保持不变。但是多了一个定义转换时间的参数 tdc_conv_cnt。

3.5.4 模式选择标准

表 3.1 模式选择标准

应用	模式		模式	描述
不要求低功耗的高精度应用	拉伸/连续	连续模式	single_conversion = 0 stretch = 0	连续测量，4MHz晶振持续开启
高精度低功耗应用		拉伸连续模式	single_conversion = 0 stretch = 2 or 3 cycle time = cytime*100μs	连续测量，4MHz晶振只有在放电时间测量期间才开启
低功耗，机械振动稳定的应用，如压力传感器	拉伸/单一转换	单一转换模式	single_conversion = 1 stretch = 0	低功耗，欠采样->不能抑制机械振动
高精度，低功耗应用，如电池驱动的3000分度的OIML电子秤		拉伸模式	single_conversion = 0 stretch = 2 cycle time = cytime*100μs	低功耗，超采样，可抑制机械振动
高精度低功耗应用，如太阳能秤		拉伸单一转换模式	single_conversion = 1 stretch = 2 or 3 cycle time = cytime*100μs	低功耗，超采用，可抑制机械振动

3.5.5 连续模式中的转换时间/测量刷新率

一次完整测量所需要的时间可以根据下面的公式计算出来。

$$T_{\text{conversion}} = \text{CycleTime} * (2 * \text{avrate} * \text{Bridge-factor} + 6 + \text{Mfake} * 2 + 1)$$

Mfake = #伪测量次数，温度测量开启

Mfake-寄存器	#伪测量次数	注释:
0	0	需要进行一些伪测量来避免
1	2	当下一次测量的时候ALU还在进行
2	4	前一次测量的数据处理
3	16	

针对上面的计算的例子:

例1: Cycle time = 110 μ s
 AVRRate=12
 四驱电桥
 Mfake=1
 $T_{\text{conversion}} = 110 \mu\text{s} * (2 * 12 * 4 + 6 + 2 + 1) = 11.55 \text{ ms} \rightarrow$
 最大测量频率为 86.6 Hz

例2: Cycle time = 110 μ s
 AVRRate=450
 全桥
 Mfake=2
 $T_{\text{conversion}} = 110 \mu\text{s} * (2 * 450 * 2 + 6 + 4 + 1) = 199.21 \text{ ms} \rightarrow$
 最大测量频率为 5.02 Hz

3.5.6 转换时间/测量频率（单一转换模式）

如果 PS081运行在单一转换模式 (Bit 4.configreg_02)，测量频率则是由configreg_00，tdc_conv_cnt[23:16]中的数值来确定的。这个值直接关系到转换的时间（乘以6.4ms）。

例如:

configreg_00: 0x158200 $\rightarrow$ tdc_conv_cnt[23:16] = 0x15 = 21 十进制
 $\rightarrow 21 \times 6.4\text{ms} = 0.1344 \text{ ms}$
 $\rightarrow$ 测量频率 = $1 / 0.1344\text{ms} = 7.44 \text{ Hz}$

注意:

如果您应用单一转换模式，需要一次完整测量的时间必须要在转换计数器(tdc_conv_cnt)中设置的时间相适合。

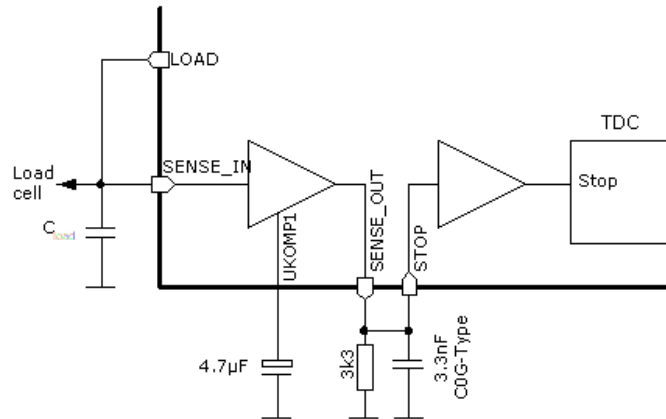
3.5.7 比较器

放电循环的结束是由一个比较器控制的。PS081是一个低噪声的内部比较器。另外，也可以选择使用外部比较器。

3.5.7.1 内部比较器

设置reg11的sel_compint为1选择内部比较器。使用内部比较器，2 mV/V 的传感器，5Hz刷新率以及开启MEDIAN 5软件滤波器，最高可以达到6万峰峰值显示分度。

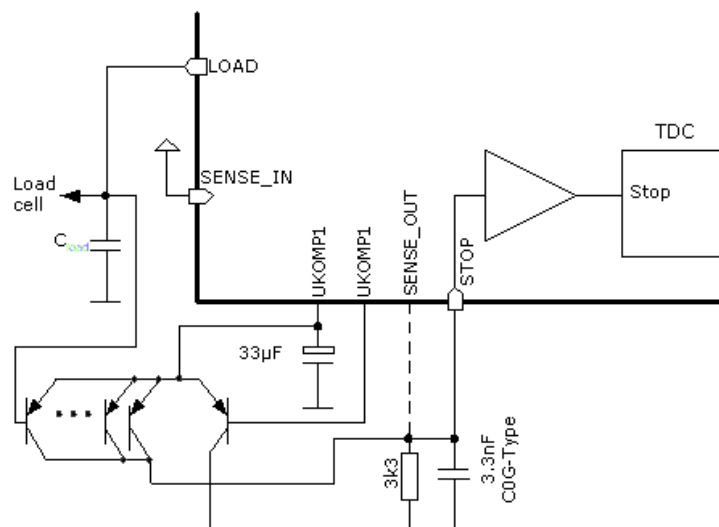
图 3.23 内部比较器



3.5.7.2 外部比较器

使用外部的双极性比较器可以更进一步的提高测量的精度，在5Hz刷新率下可以精度提高到15万显示分度。

图 3.24 外部比较器



建议：

使用低噪声PNP三极管如2N5087、CMKT5087、BC859。在LOAD端应该使用5个并联的三极管。无需对三极管进行匹配。应用COG类型的电容来做低通滤波电容。

3.5.7.3 什么时候需要使用外部比较器

a) 很高的测量精度

需要很高精度的应用，例如计数称，高精度的OIML秤。

b) 极低功耗

需要很低功耗的应用，例如太阳能衡器。由于噪声低，对于同一分辨率，AVRat的值比使用内部比较器时要小，工作电流相应地也就被降低了。使用双极性比较器时，工作电流几乎可以减小一半。

c) 超低电压

需要使用<2.1V的Vcc的应用，例如使用纽扣电池供电。使用双极性比较器将会带来非常好的效果

3.5.7.4 比较器的控制

比较器可以一直开启也可以仅在测量的时候开启（con_comp[1:0]），仅在测量时开启可以降低功耗。另外，内部比较器的工作阻抗可以通过（sel_compr[1:0]）来改变。

我们推荐使用下面的设置：

```
CON_COMP    = 'b10  → 测量时开启
SEL_COMPR    = 'b10  → 选择7K电阻
```

如果CON_COMP被设置为‘b11（持续开启），比较器需要大约130 μA@ 3.0 V持续电流。

UCOMP1 and STOP端的电容：

在 UCOMP1和STOP端的电容器对于低噪声来说是非常重要的。为了达到最佳效果我们推荐CUCOMP1=33 μF和CSTOP=3.3 nF。Cucomp1可以用普通的电解电容。但是，CSTOP推荐使用COG类型的电容。

如果使用内部比较器的话CUCOMP1和CSTOP必须与4.7k欧姆的电阻连接。不过，使用容值小一些的电容也是可以的。

推荐值：CUCOMP1: 不小于1 μ F

CSTOP: 低于CUCOMP1的1/3000

例子：CUCOMP1=1 μ F $\rightarrow$ CSTOP<1 μ F/3000 $\rightarrow$ 选择330 pF

与使用CSTOP推荐值3.3nF相比，使用330 pF的CSTOP时噪声大概会增加0.3-0.5位。

3.5.7.5 比较器延迟的修正

因为比较器的主要作用是降低噪声，所以就引入了延迟时间，而这个延迟时间是不能够被忽略的。这个延迟时间受温度影响，并且会引入增益误差，这个误差对于衡器应用的精度有非常大的影响。在应用中，用Rtemp和Rref电阻对延迟时间进行周期性测量。PS081用测量出的延迟时间来修正测量结果。

比较器的延迟时间受CUCOMP1和 CSTOP的电容值影响。因为这些值可以被用户改变，所以可通过寄存器 Mult_PP[7:0]来调整修正程式。

与推荐的CUCOMP1和CSTOP(33 μ F 和 3.3 nF)相适应的Mult_pp因数值为1.28 (0xA3, reg10)。如果电容的值增加那么修正值 Mult_PP也必须增加，反之亦然。如果选择的 Mult_PP值太低那么增益误差将会随温度的升高或者电压的下降而降低。

在选择了正确的Mult_PP值之后电路的增益将会在一个非常宽的电压和温度范围内保持稳定。增益误差的温度漂移小于1ppm/K。电源抑制比(PSRR)大于130 dB。请参见章节3.6.2。

3.5.8 温度测量

PS081提供了温度测量的可能，接一个外部的温敏电阻，就可以测量温度了。

温度信息可以用来修正非补偿压电元件的增益漂移，我们称之为“Rspan-By-Temp”补偿。

温度测量通过测量两个电阻(Rtemp and Rref)的放电时间的比率来完成的。其中一个为温敏电阻，另一个为非温敏电阻。比例的变化将回在内部执行的硬件算法调用来修正非补偿传感器的增益漂移。当然，此电阻对的温度系数需要与传感器的温度系数相匹配，这个可以通过使用参数TK_Gain(configreg_08)来完成此匹配，在应变测量之前先修正温度测量结果。

mod_rspan和rspan_by_temp必须置1(configreg_01, Bit6和8，软件评估页的ALU页中)。另有单独的文件来说明如果确定合适的TK-Gain值，如需要更多信息请联系acam。

3.5.8.1 Rtemp和 Rref电阻

敏感电阻可以用炭膜电阻200-300ppm/K, 参考电阻可以用金属膜电阻, 一般典型为<50ppm/K。请注意, 不要应用<200ppm/K的碳膜电阻, 否则温度测量将不能正常进行。

Rtemp和Rref的值必须要随着应变电阻的阻值和桥的连接方式来进行调整。因此, 电阻需要为以下阻值:

一般: (半桥、全桥、四驱桥)	$R=R_{sg}$	(例如: 1000 Ω 对应1000 Ω 的电桥)
惠斯通桥:	$R=0.75R_{sg}$	(例如: 750 Ω 对应1000 Ω 的电桥)

注意:

只要需要修正比较器的延迟时间, 就必须连接这两个电阻。如果不进行温度测量, 这两个电阻可用同一类型的电阻 (例如: 炭膜电阻)。

更多的增益误差纠正方法见说明说应用指南AN021

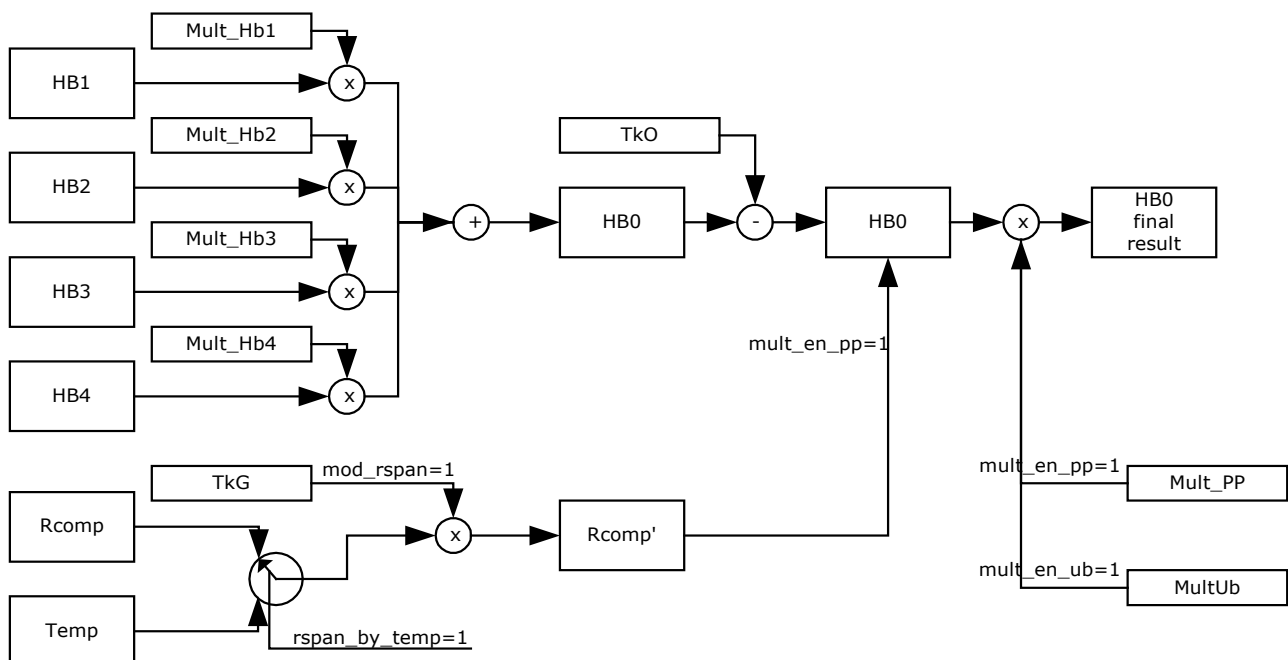
3.6 预处理

在测量结束的时候，转换器调用ROM里的程式对测量进行一些预处理。它把已校准和已定标的结果存入RAM的结果寄存器中。之后，如果epr_usr_prg =1，那么EEPROM 中的程序将被启动。

预处理的特性:

- 4 个半桥的结果是有各自独立的乘法因数。这使得用软件修正四驱电桥应用中的角差称重成为可能。
- 应变传感器和span 补偿电阻是分开的。因此可通过软件来调节增益补偿电阻。甚至可以用温度测量代替span 补偿电阻。使用这种方法，仅采用软件就可以将普通传感器的性能提升到高质量传感器的性能。
- 相应地，使用PICOSTRAIN做转换器，称重系统的漂移就不受span补偿的影响了。使用软件就可以修正这个漂移。
- 被修正过后得到的结果还会受到电池电压不稳的影响，因此还需要再乘以修正电压影响的修正因子。这有助于抑制电源不稳定带来的影响，在此情况下允许在不进行电压调整情况下直接用电池供电。

图 3.25 预处理



一个简单的显示测量结果的程序如下：

```
ramadr      20          ; HB0结果
move        x , r       ; 加载测量结果到x-累加器
move        y , 2       ; 小数点的位置到 y-累加器
no2lcd      ; 7-段码的转换
newlcd      ; 刷新LCD驱动
clrwdt      ; 将看门狗定时器重置
stop        ; 停止 uC工作
```

3.6.1 四角秤的角差补偿

一些衡器例如人体秤是需要4个传感器的，每一个传感器有一个半桥。如果每个传感器的灵敏度不是完全一致的话，重量放在秤体不同的位置将会导致不同的输出结果产生。PS081可以通过软件无需任何硬件措施来进行这个角差的补偿。每一个半桥都有自己的乘法因子（Mult_HB1到Mult_HB4）。通过4个简单的测量就可以获得四个分别的补偿系数。因此在补偿测量中需要将重量分别放到秤的4个角上。如果您想应用四角补偿系数Mult_HB1到Mult_HB4, 请与acam公司联系。

3.6.2 传感器的Gain 和 Offset 漂移的温度补偿

现今高端的转换器零点漂移和增益漂移性能是非常好的。那么转换器本身的漂移比一个好的传感器漂移要更优秀5-10倍。那么最终的目标是将整个系统（整秤）的温度漂移性能提高而不仅仅是一个好的温度特性的电路部分。通过acam公司的Picostrain系列产品，可以提供给您一种方法，在不改变传感器硬件结构的基础上仅通过软件来矫正整个系统的零点和增益偏移。这个方法仅适用于只有一个桥臂有温补电阻（Rspan）的传感器上。¹

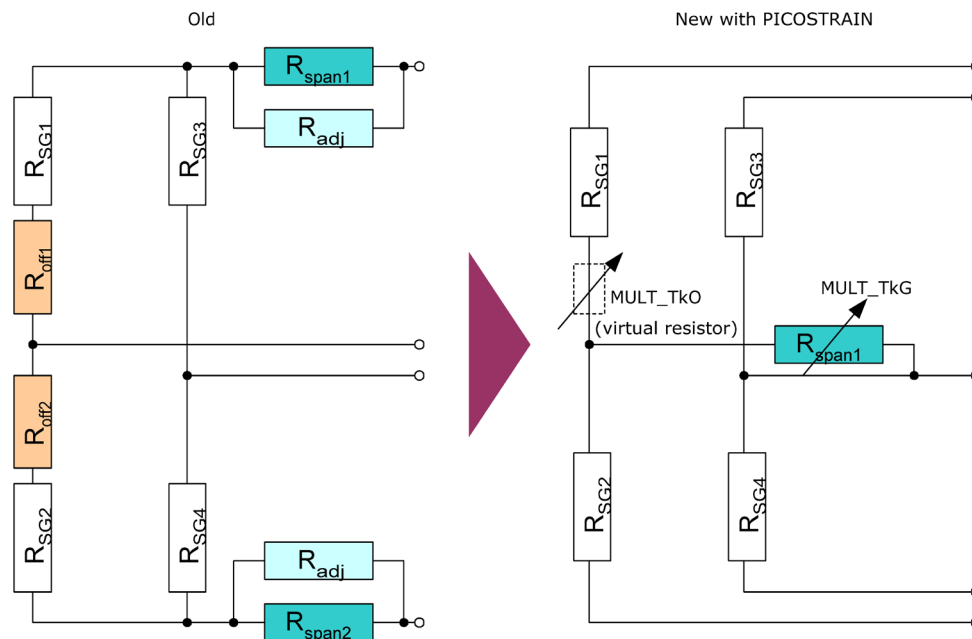
如果传感器上仅有一个桥臂带有补偿电阻（Rcomp或Rspan），PS081就可以测量这个阻值，并通过内部单片机中的算法和因数MULT_TkG增益补偿系数以及MULT_TK0零点补偿系数来纠正传感器温度特性。没有必要使用精确的补偿电阻。也没有必要对Rcomp进行手动的硬件修正。

还有一个好处就是可以把offset与gain分离，这样传感器上就不再需要offset补偿电阻。PS081可以通过MULT_Tk0来修正offset。

通过这些，PS081改善了传感器的增益和零点漂移性能。若不想改变传感器和电路，但是又想提高整个衡器的测量精度，这种方法是非常合适的。特别是数字传感器，整个传感器生产可以被简化并能大大降低成本。

¹ 请参看章节 3.6.4 温度增益偏移非线性度。有可能需要给Rspan温度补偿电阻加一个并联电阻来降低非线性度。

图 3.26 增益和漂移补偿的修正



下面和一些应用MULT_TKG和MULT_TKO补偿的例子：

- 如果补偿电阻与传感器相匹配，但是桥本身有偏移，这个偏移是可以通过软件来消除的。
- 如果是在温度漂移箱里进行测试，就能够确定十分合适的修正系数MULT_TkG和MULT_TkO。在这个情况下整个系统的补偿得到了显著提高。采用这种整个衡器都装配好后的后修正方法，整个衡器的偏移和增益可以被调整到几乎完美的情况，其性能甚至比高端衡器要求的还要高很多。（例如：整个衡器增益<1ppm/K，漂移<10Nv/K）。
- 如果传感器的增益误差已知（比如在整个批次增益误差是一定的，但是是错误的值）那么可以直接通过PS081进行修正，且无需在恒温箱中进行修正测试。

Acam公司对于传感器的温度补偿写了一份详细说明，见白皮书WP002，在白皮书中阐述了补偿的方法以及重要性。另外这个白皮书一步一步的指导您如何通过Mult_TkO和Mult_TKG来补偿温度。

PS081同样也可以补偿不带补偿电阻的传感器（传感器桥臂没有任何补偿电阻Rspan）。那么这个情况下取代Rspan的是应用板上温度测量的信息。这个模式是通过设置寄存器1中的Rspan_by_Temp为1来开启的。另外这个调整也是通过系数MULT_TkG和MULT_TkO来完成的。那

么这个补偿的精确程度取决于在温敏电阻的 T_k 和其精确度。一般来说对于未补偿的传感器通过这种方法可以提高温度性能至少6-8倍。这个补偿一般对于普通的商用和家用秤应该足够了。对于高精度的满足度量学标准的衡器，我们推荐应用传感器上的Rspan电阻以及这里所描述的Mult_TKG和Mult_TKO方法进行补偿。

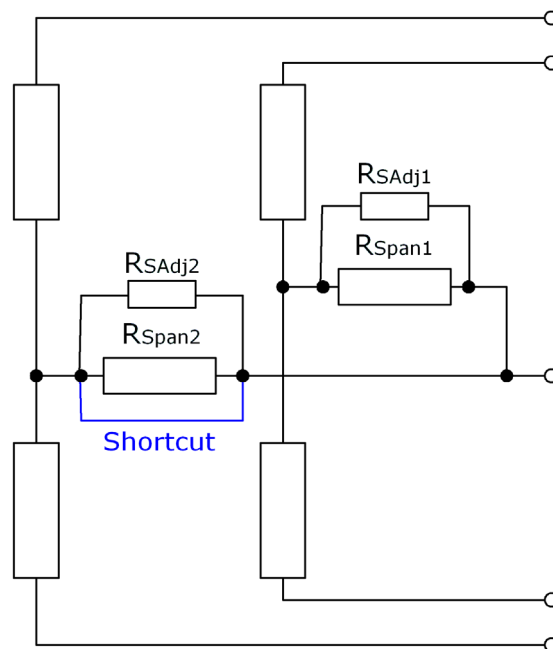
在应用指南AN021当中，有对于应用2个电阻这种方法（Rspan_by_temp）的更加详细的介绍。

3.6.3 Rspan的一些注释

PICOSTRAIN测量方式仅仅需要1个Rspan的温度补偿电阻。由于PICOSTRAIN 产品的共模抑制比(CMRR)接近于无限(高达135dB)，并不需要应用2个Rspan。而且，PICOSTRAIN实际上也不能够处理带有两个Rspan电阻的全桥。

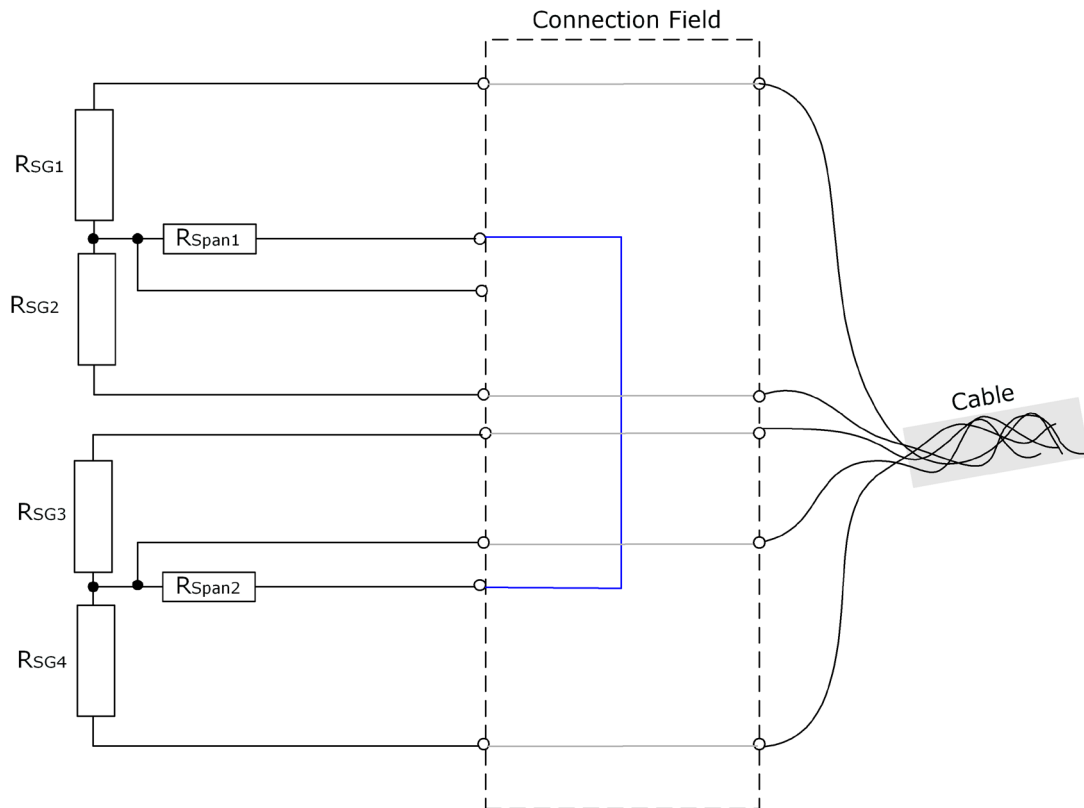
那么最简单的则是应用本身仅带有一个1 Rspan电阻的传感器。如果你希望首先应用带有两个Rspan电阻的传感器进行测试，你可以将其中一端的Rspan电阻如下进行短接：

图3. 27



一个更方便的方法就是将传感器的2个Rspan电阻进行串联起来。如果传感器的Rspan电阻如图所示那么就可以进行这样的连接操作：

图 3.28



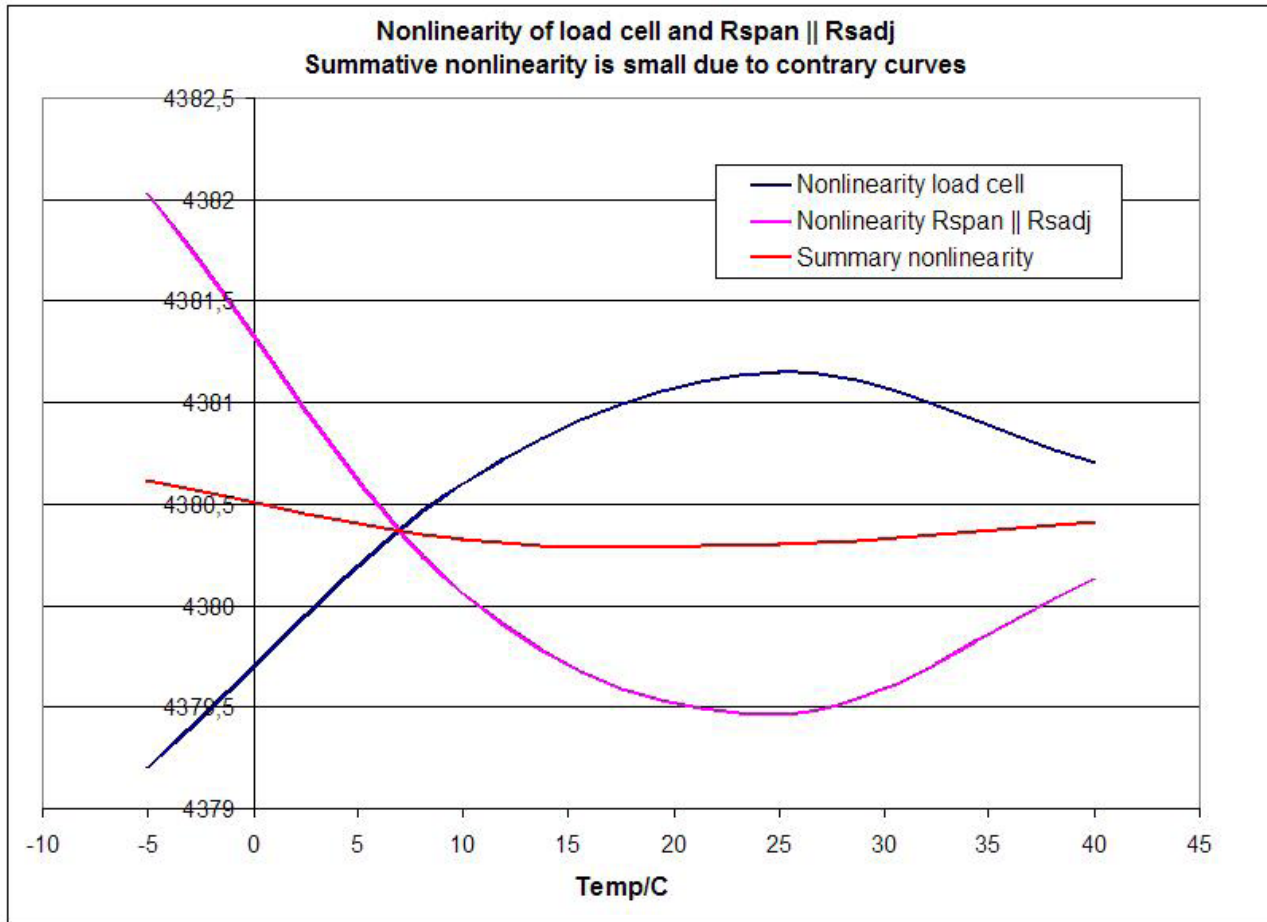
请注意：你并没有直接接触到传感器的应变或者Rspan补偿电阻。你仅是在传感器的前端面板对于走线进行改变。这个同样对于Rspan的改变以及从惠斯通连接到PICO STRAIN连接是有效的（请参看章节 3.3.2）。

3.6.4 温度增益漂移的非线性度

此章节涉及的范围： 对于商用秤比较重要，例如需要满足OIML标准。

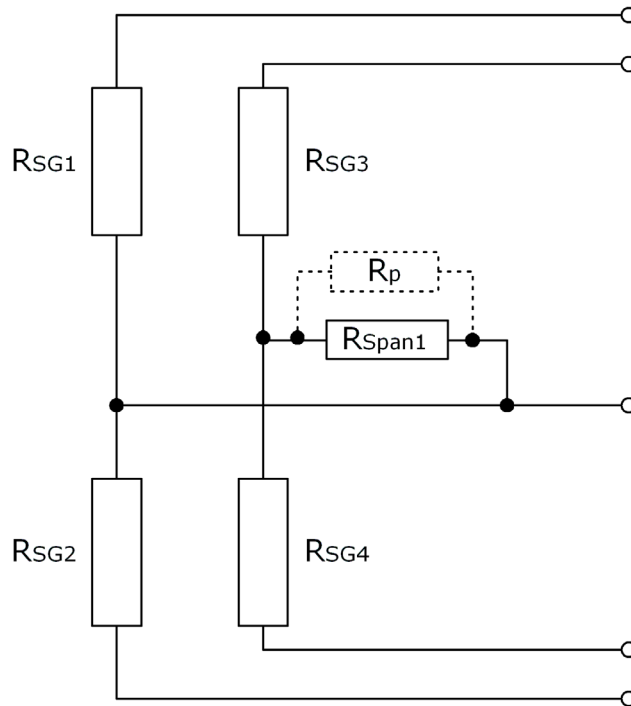
与 PICO STRAIN 增益补偿无关，传感器都会有一个对于温度的非线性度。这个非线性度有两个来源，一个是传感器自身的非线性度（材料，胶，连线等等。）而另一个非线性度则由于并联的Rspan温度补偿电阻和其并联的调整电阻（R_{adj}）引入。一般来说这两种非线性度的表现的方向是正好相反的，因此整体的非线性是被消除的。换句话说，给Rspan并联调整电阻以及传感器本身的非线性度是相互抵消的。请参考下图说明了这个效应：

图 3. 29



那么如果当这个调整电阻 (Rsdj) 不存在的情况下那么整个性能将会被改变，那么对于非线性的补偿效果将会消失。在PICOSTRAIN的最基本set-up当中，传感器仅带有1个Rspan电阻，没有任何并联的调整电阻，这个电阻是不存在的。这个在传感器自身的非线性度很好的情况下是没有什么问题的。但是如果传感器本身的非线性度不能被忽略的话，需要在补偿电阻上加一个并联电阻来补偿这个非线性度。请注意，这个并联电阻 (Rp)的目的并不是出于纠正Rspan补偿电阻的阻值而加，而仅用于非线性度的补偿。

图 3.30



有几个标准来决定是否要加这个并联电阻(R_p)。详细的信息请参考我们的技术白皮书WP002。对于是否推荐加这个电阻 R_p 的一些建议以及自动计算这个电阻阻值的Excel表格可以通过acam公司获取。如果推荐加这个 R_p 电阻,那么这个电阻的类型可以是普通的固定电阻无需特殊类型。

3.6.5 PS081自身的Gain漂移—通过MULT_PP进行优化

PS081的增益漂移很低,在参数MULT_PP设置适当的情况下,可以小于1ppm/K。PS081产生增益漂移的原因与AD转换器是不同的。因此,我们在这个章节中介绍一下PS081的增益漂移产生的原因,同时也介绍一下如何进行正确测量。

背景:

在典型的A/D转换器应用中,运算放大器的电阻的温漂需要进行非常精确地匹配。如果不匹配的话,会有比较大的增益漂移。而PS081的内部构造则是完全不同的。PS081使用的是时间数字转换器(TD-Converter),不需要使用前置放大器。

PS081电路的增益漂移主要来自比较器的时间延迟。在惠斯通模式中,模拟开关是在这个延迟中的一部分,因此选择不同型号的模拟开关会影响这个数值。当设置寄存器2的pptemp为1的时候,PS081可以通过在PSEP1和PSEP2端口的两个电阻测量这个延迟。那么这样可以通过系数MULT_PP来进行补偿。

这个延迟时间取决于和比较器晶体管中的集电极并联的低通滤波电容的数值。我们推荐集电极电阻使用 $3.3\text{k}\Omega$ 电阻和并联电容使用 3.3nF 的电容。如果电容的值改变了，那么最佳的MULT_PP因数也随之改变。使用内部比较器还是使用外比较器会影响MULT_PP因数，还有就是硬件的改动也会影响到MULT_PP的设置。

通过Mult_PP进行增益误差的优化

通过我们的硬件推荐之后，如果设置Mult_PP为1或者不应用这个系数的话那么整个系统的增益误差大约为 8ppm/K 。那么这个误差可以通过设置正确的Mult_PP的值进一步降低到小于 1ppm 的范围之内。

一旦在开发过程中确定了Mult_PP的数值，这个数值可以应用到整个生产当中。绝对没有必要对于每一个电路进行Mult_PP的确定！

注:

PS081本身的增益偏移是非常小的接近零值。主要的增益偏移是由于非线性部分以及比较器的延迟时间所引起。这个非线性部分本身是由于在比较器第一阶部分的低通滤波所引起（ $3.3\text{k}\Omega$ ， 3.3nF ）。这个可以通过Mult_PP非常显著的进行补偿。因为这个低通滤波可以非常精确的来确定并且在整个生产过程中非常稳定无需再调整。

如果使用我们推荐的硬件，我们确定的MULT-PP数值如下：

- | | |
|--|------|
| 1. 惠斯通电桥方式应用外部比较器以及模拟开关为TS5A3160 (TI) | 1.28 |
| 2. 惠斯通电桥方式应用内部比较器以及模拟开关为74LVC1G3157 (TI) | 1.14 |
| 3. PICOSTRAIN标准连接，外部比较器，我们的推荐值 | 1.28 |

对于惠斯通桥我们推荐使用TS5A3160因为这个器件在低电压小于 2.7V 的时候有一个非常好的性能，那么74LVC1G3157也同样是一个不错的选择，但更适用于 3V 电压或者更高的电压。如果我们所推荐的硬件设置被改变，尤其是当比较器部分的低通滤波被改变的时候，Mult_PP数值必须要进行重新确定。这个硬件设置的数值是对于整个生产所适用的，而且是与生产批次没有关系的。详细的信息请参见应用指南AN018。

3.6.6 PS081本身的零点漂移

PS081的零点偏移与AD数模转换的原因也是不同的。那么PICO STRAIN原理的零点偏移主要是由于路径的寄生电阻没有或者没有被完全补偿的结果。在PS081的封装版本芯片中主要是由于绑定线的长度不同所造成的。

由于这个零点偏移的存在，漂移的数值属于应变电阻有关的。应变电阻的阻值越小，那么存在的漂移越大。比如1k欧姆的应变电阻漂移数值为350欧姆应变电阻漂移的三分之一。

对于最好的漂移特性我们推荐使用标准的全桥连接。系统的零点漂移在这个模式下为大约 $\pm 10 \text{ nV/V/K}$ 。这个模式是满足OIML标准的50%，PS081可以应用在高端的衡器当中（参见应用指南AN018）。

在所有的PICO STRAIN模式中传感器的连线的电阻为零点漂移的一部分。为了降低这一部分的偏移，请仔细观察一下从传感器到电路部分的长度。那么一般来说最主要的部分是PCB，几毫米的不匹配将会在offset中体现的非常突出。而传感器线并不是最主要的因素，因为线的直径是很大的。

那么惠斯通桥连接则是一个特殊的情况。在这个模式下由于这种测量方式几乎100%的寄生电容是被补偿的。因此PS081的惠斯通模式的零点非常接近0值，而且尽管导线不匹配也可以被补偿到小于1 nV/V/K。

作为比较：

- 为了能够满足 OIML 3000 的标准，整个秤的零点漂移不能够超过133nV/V/K
- 为了能够满足OIML 10000的标准，整个秤的零点漂移不能够超过40nV/V/K

下面的表格给出了PS081在不同的连接情况下的一个典型offset的值。为了获得理想的最大值和最小值我们给每一个典型值乘以系数3。那么您会对于整个生产批次的情况获得一个大概的分布情况。（非保证）。

在不同模式下的典型offset值

模式	350 欧姆应变电阻	1 欧姆应变电阻	OIML 10000
全桥标准连接 *	$\pm 10 \text{ nV/V/K}$	$\pm 4 \text{ nV/V/K}$	$\pm 40 \text{ nV/V/K}$
惠斯通桥连接	$< \pm 1 \text{ nV/V/K}$	$<< \pm 1 \text{ nV/V/K}$	$\pm 40 \text{ nV/V/K}$

*在PCB上做路径匹配，请参见如下解释。

全桥标准连接的零点offset偏移

在PS081的封装版本中，acam公司必须要确定，那么种模式是最优模式。我们选择了全桥并联模式，因此由于绑定线长度的不同，在全桥标准模式中有一个系统的零点偏移在里面，对于每一颗芯片都会有相同影响。这个零点偏移为 $\pm 10 \text{ nV/V/K}$ 。由于这个偏移的特性已经确定我们可以在PCB上通过匹配不同长度的绑定线来对其进行补偿。

为了补偿这一系统零点漂移，在SG-B1到连接传感器端口的路径应该比SG-A2到连接传感器端口的路径长（待定）mm。而SG-A1和 SG-B2 应该有同样的长度。

请同样注意:

无论在任何情况下，主要的零点偏移都不是由于电路部分所引起的。PICO STRAIN以及好的AD-转换器的零点偏移值都要比一个好的传感器的零点偏移值优秀5 - 10倍。因此，如果要大大提高系统的性能，那么必须要降低传感器本身的零点偏移。PS081可以通过软件调整MULT_TK0系数来完成传感器的零点补偿！

3.6.7 Mult_UB – 电源偏移抑制

PS081可以非常频繁的对电压进行测量。这个测量的电压可以应用在矫正由于电压变化而引起的增益偏移上。可以通过设置寄存器位mult_en_ub为1来开启这一功能。系数mult_ub(7:0)定义了电压测量的控制比率。一般来说控制比率是非常小的。应变测量结果应该根

据下面公式进行校正：

$$HB = HB / (1 + UB * [-128 \dots 127] / 221).$$

对于Mult_UB的一个标准设置为 0xF7。

目录

页面

4	时钟元件 & 特殊设置.....	4-2
4.1	晶振.....	4-2
4.2	内部LCD驱动	4-3
4.2.1	基本配置	4-3
4.2.2	LCD 供电电源.....	4-4
4.2.3	LCD 驱动方法.....	4-8
4.2.4	LCD 控制	4-9
4.2.5	连接原理图	4-10
4.2.6	设置各段的位置	4-12
4.3	对于外部LCD驱动的支持.....	4-14
4.4	I/O 管脚	4-16
4.4.1	配置	4-16
4.4.2	端口定义	4-16
4.4.3	输出-写操作.....	4-17
4.4.4	增加I/O个数	4-17
4.4.5	输入 - 读	4-18
4.5	SPI 接口	4-19
4.5.1	接口	4-19
4.5.2	SPI 时序.....	4-19
4.5.3	SPI 指令.....	4-20
4.5.4	外部单片机通过SPI端口控制PS081 (非-连续 配置)	4-23
4.6	电源电压.....	4-25
4.6.1	滤波 / 推荐的 LDO.....	4-27
4.6.2	电压测量	4-28

4 时钟元件 & 特殊设置

4.1 晶振

PS081有一个内部10KHz的低功耗晶振，此晶振起到了基础计时器的功能，并且在拉伸模式的周期和测量范围1中用来确定周期时间。

另外，PS081还带有一个晶振驱动器，用来驱动外部4MHz陶瓷振荡器。这是用来测量时间和定义测量范围2中的周期时间的。其功耗大约是130 μ A@3.0V。

配置:

- 寄存器3，位17-19: sel_start_osz
- 0 = 关闭晶振
- 1 = 晶振持续运行
- 2 = 在打开晶振延迟100us后开始测量
- 3 = 在打开晶振延迟200us后开始测量
- 4 = 在打开晶振延迟300us后开始测量
- 5 = 在打开晶振延迟400us后开始测量
- 6 & 7 不连接
- 寄存器2，为0: 自动使用10kHz晶振

晶振可以设置切换为持续工作或者只在测量期间工作。包含一些启振时间以使晶振达到满振幅（sel_start_osz[2:0]）。4MHz晶振的启动时间与供电电压有一定关系，大概为50us到100us。

自动校准:

内部的10KHz晶振可以通过4MHz晶振来自动校准。温度和电压的频率变化，将影响更新速率和采样速率，因为10KHz晶振是时间数字转换器的基础时钟，也是拉伸Stretched模式的周期。建议使用自动校准选项设置为Atuo10K=1。

注:

自动校准不能用于拉伸Stretched单一转换模式。

布线推荐:

晶振需要放在PS081的附近的地方。晶振四周必须环绕铺地层。SPI接线不能跨接晶振的连线。4M晶振需要有一个1M欧姆的下拉电阻来避免当关闭晶振时的跨越电流产生。这个电阻大大降低了晶振的电流消耗，因此在太阳能应用中必须被使用。

4.2 内部LCD驱动

LCD驱动有如下特性：

- 18个管脚为
 - 1/4占空比最多6 个数字包括小数点和8个特殊字符
 - 1/3占空比最多5 个数字包括小数点和5个特殊字符
 - 1/2 占空比最多4个数字包括小数点
- 将显示器的电压稳定在3V, 2.5V和2V
- 为3V和2.5V LCD集成的电压倍压
- 低功耗的2V操作无需电压倍压
- 可工作在非稳压电源下如锂电池或者太阳能电池
- 休眠状态下电流消耗非常小
- 驱动强度可以根据段码尺寸以及电流消耗进行调整
- 输出可调整，因此可以连接已经连好的显示器
- 嵌入式的7段码数字显示的转换表格
- 嵌入式的24数字转换的ROM代码

4.2.1 基本配置

开启带有lcd_duty的LCD然后设置为特定的multiplex模式

```

lcd_duty    = 0    关闭
             = 1    2x multiplex
             = 2    3x multiplex
             = 3    4x multiplex 模式
  
```

lcd_freq 控制了所有像素的开启时间。一个像素开启时间越长那么需要越少的电流。对于无闪烁的显示来讲推荐使用刷新率 > 30 Hz。因此开启时间则取决于所选择的multiplex模式。

lcd_freq[2:0]	=	像素 开启时间	Multiplex 的模式		
			1/4	1/3	1/2
0		8.0 ms	15	20	31 Hz
1		4.8 ms	26	34	52 Hz
2		4.0 ms	31	42	62 Hz
3		3.2 ms	30	52	78 Hz
4		2.4 ms	52	69	104 Hz
5		2.0 ms	62	82	125 Hz

4.2.2 LCD 供电电源

PS081内部集成一个充电泵来加倍和稳定驱动3V和2.5V LCD显示。2V的显示可以很容易的直接通过电源进行供电。外部电容的选择要根据LCD的尺寸或者其电容大小来决定。

配置:	寄存器 11: lcd_standby, lcd_vlt 寄存器 16: en_noise_lcd, lcd_direct_drive, use_10kHz_lcd 寄存器 17: lcd_pulsed, c10_div_lcd
lcd_standby	LCD有一个休眠模式。在这个模式里面LCD是被关闭的，但是电压产生被转换到高阻抗状态。因此可以非常快的将显示屏开启。这在自动开启模式中也许是一种不错的选择。 = 0 LDC 开启 = 1 休眠
lcd_vlt[1:0]	LCD 供电电压选择 = 0 2.0 V = 1 2.5 V = 2 3.0 V = 3 2.0 V 不带电压倍压
lcd_directdrive	LCD 驱动选择 = 0 关闭 = 1 LCD直接被Vcc驱动，不带充电泵和电压调节。这将减少LCD电流，应用在太阳能驱动。
lcd_dis_chargm	1 = 抑制在测量期间充电泵电容进行再充电 这个选择只适用于拉伸Stretched模式（推荐）和单一转换模式[在内部低平均率（avrate小于10）测量之间的间隔时间足够用来电容充电。duty 1: 3]。 这一选择只在与4MHz晶振脉冲同时使用的情况下工作。
use_10kHz_lcd	1 = 选择10KHz的低压电源做为LCD充电泵。 1 = 推荐 0 = 充电泵时钟来自4MHz，分频数可以设置的范围480-512。这一选择避免了充电泵和测量之间的耦合关系，因此降低了测量失真发生的可能。

c10_div_lcd[5:0] 5位的分频系数来从4M晶振持续模式中给出10khz的频率。

分频系数 = $448 + c10\_div\_lcd$

lcd_pulsed[3:0] 自动进行脉冲的LCD显示，尤其是太阳能应用中的休眠模式定义了LCD开启的周期，以及LCD启动的时间。

00xx 300 ms 闪烁周期

01xx 600 ms 闪烁周期

10xx 1 s 闪烁周期

11xx 2 s 闪烁周期

xx00 关闭 启动时间

xx01 100 ms 启动时间

xx10 200 ms 启动时间

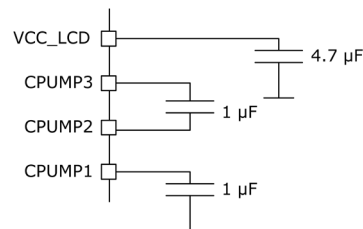
xx11 400 ms 启动时间

en_noise_C10_lcd = 0 关闭

= 1 给在C10_div_lcd中的分频系数加噪声

4.2.2.1 带有电压加倍器时2.5V和3.0V的操作

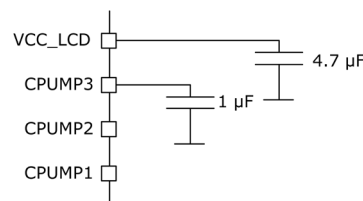
图 4.1 带倍压的充电泵



第一步两个电容都被充电到一半的显示电压，1.5V或1.25V。这个电压可以在管脚CPUMP1和CPUMP3被检测到。第2步两个电容被改变成串联接法。管脚CPUMP3将显示全部LCD电压，而管脚CPUMP1和CPUMP2显示了LCD的一半电压。

4.2.2.2 不带电压加倍时2.0V的操作

图 4.2 不带电压倍压得充电泵



在这个模式下LCD电压被从一个不稳定的电源电压通过电容的充电稳定到LCD电压。供电电压在这个模式下不能

够降到2V以下。

4.2.2.3 直接驱动

第3个选择是在没有对电压调整的情况下直接通过电源进行供电。因此不需要外部电容连接，输出被设置为低阻抗输出

```
lcd_r_const      = 0    (10k0hm)
lcd_vlt[1:0]     = 0    2.0 V
```

然而，在CPUMP1和CPUMP2上加上一个1uF的电容有助于降低噪声，如图4. 2所示。

在直接驱动模式中推荐使用的LCD刷新频率越低越好，例如：lcd_freq=1。

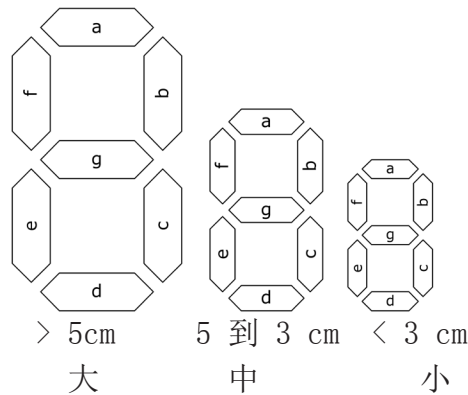
直接驱动模式是通常推荐的驱动模式，用来驱动带有内部LCD-驱动 LCD。

4.2.2.4 LCD输出驱动配置

输出驱动的内部电阻可以根据显示器的尺寸进行调整。通过这个方法可以进一步将电流消耗优化。显示器的尺寸影响了：

- 内部驱动电阻
- 充电泵的内部最小充电时间
- 显示器重新刷新的时间

图 4. 3 LCD 段的尺寸



配置寄存器：

控制位	大	中	小	功能
lcd_fastld[1:0]	3	2	1	配置了快速充电的周期数(10ms) 带有低欧姆分压器
lcd_swloadlk	1	1	1	0 = 通过200欧姆电阻给电容充电 1 = 通过1K欧姆电阻给电容充电

lcd_r_const[1:0]	1	2	3	设置了LCD驱动的跨接电阻值 0 = 30 k 1 = 50 k 2 = 200 k 3 = 800 k
lcd_charge[1:0]	0	2	3	在重新充电前选择需要等待多少个LCD时钟周期 0 = 每个周期 1 = 两个周期 2 = 4个周期
	0	0	0	在Stretched拉伸模式必须设置为0，与直接驱动模式无关
lcd_r_fastld	3	2	1	配置快速充电周期（10ms） 带有低欧姆电阻分压器

4.2.2.5 LCD充电泵和测量串扰

LCD充电泵因为串扰可能会对测量质量造成负面的影响。比较典型的，这会使测量结果产生明显的周期性振动。

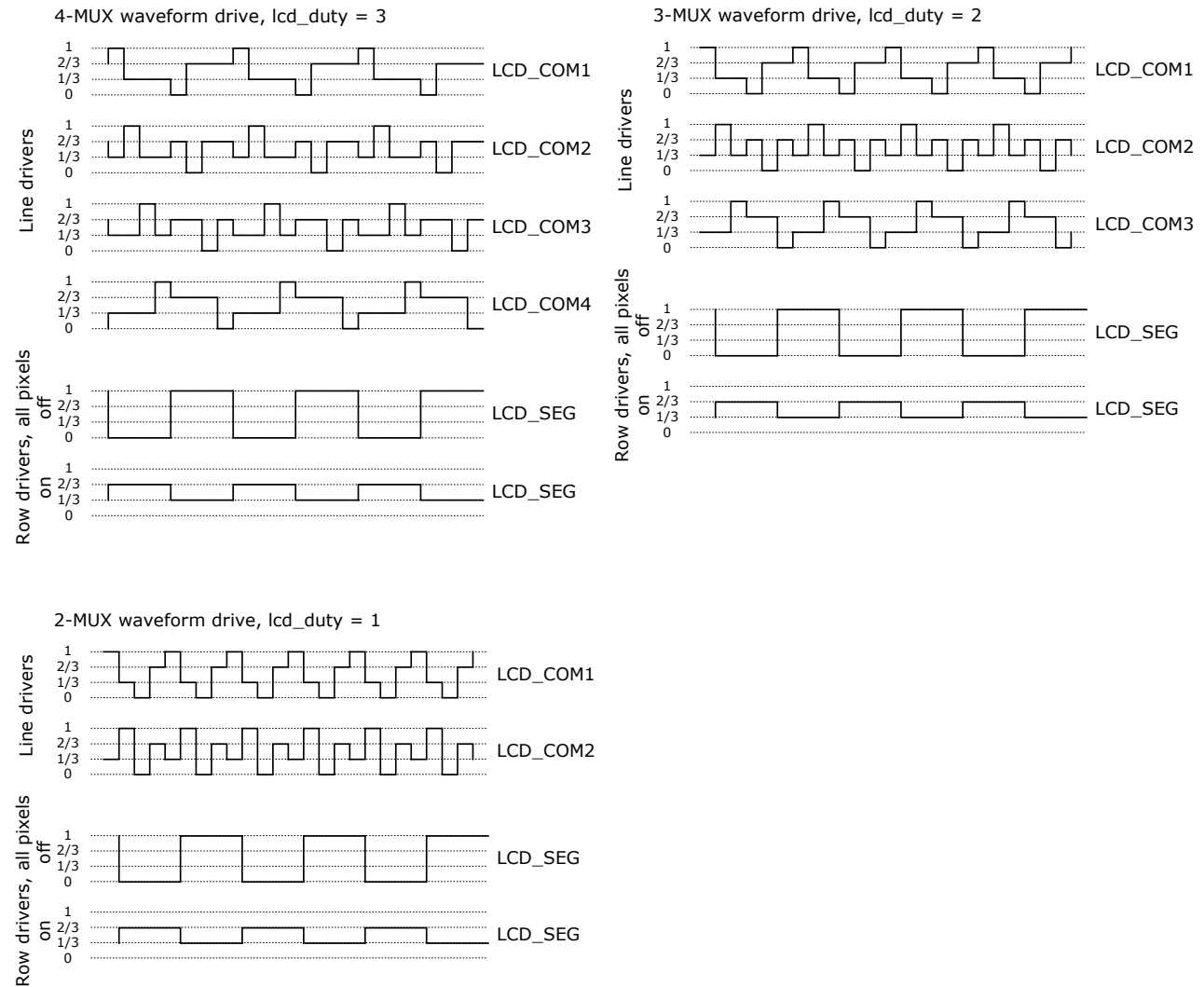
PS081提供了若干配置用于克服这个现象。接下来我们展示一些测量。这些不是唯一的解决办法，但是是一个选择正确应用的需要。任何情况下不确定的问题请及时联系acam获取支持。

- 在直接驱动模式中，LCD不影响测量。
- 在很多情况下，选择正确的LCD频率可以解决问题。总的来说，频率越高失真越小。测试不同的lcd-freq值可以得到最好的选择。功耗可能相应增加一点。在电池供电系统中这不是问题。任何情况下太阳能应用中我们都推荐直接驱动模式。
- LCD驱动电压电阻阻值要尽可能的大。这可能与测量2冲突，可以看到部分将关闭。正确的选择必须要在实验中得到证实。
- VCC_LCD上的滤波电容可以加大。我们推荐的标准是4.7uF。如果有问题的话加大到22uF能明显得到改善。

4.2.3 LCD 驱动方法

在每个输出驱动模式中都有4个等级，0、1/3、2/3和全LCD电压。

图 4.4 LCD 驱动



4.2.4 LCD 控制

一个数字由8个部分组成，每个段是由从a到g的字母命名。小数点用h表示。

每一部分通过设置寄存器13、14和15的相应位到“1”或者“0”来开启或者关闭。这个分配与multiplex模式无关，如下图所示：

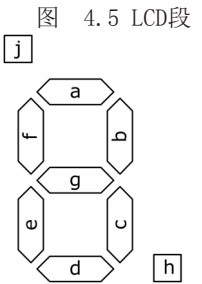


表 4.1

数字	段		Hex 值	在配置存储器中的位置： 在 2x multiplex 模式下，使用的是lcd_segment的低32位。 在 3x multiplex模式下，每一个数字都是以一个3x3的矩阵表示的，包括一个额外的特殊字符。lcd_segment的低40位用于控制5个数字。特殊字符是通过位40到44控制的。
	hgfe	dcba		
“0”	0011	1111	3F	
“1”	0000	0110	06	
“2”	0101	1011	5B	
“3”	0100	1111	4F	
“4”	0110	0110	66	
“5”	0110	1101	6D	
“6”	0111	1101	7D	
“7”	0000	0111	07	
“8”	0111	1111	7F	
“9”	0110	1111	6F	

表 4.2

数字	lcd_segment	寄存器配置	应用
6	[55:48]	15	1/4
5	[47:40]	14	1/4, 1/3
4	[39:32]	14	1/4, 1/3
3	[31:24]	14	1/4, 1/3, 1/2
2	[23:16]	13	1/4, 1/3, 1/2
1	[15:8]	13	1/4, 1/3, 1/2
0	[7:0]	13	1/4, 1/3, 1/2

通过dez2lcd (D)，单片机里面有一个特殊操作码可以将十进制数据转换成0-9字符。它将所寻址的累加器的最低4位代码转化成标准7段码。

为了进一步更方便，在ROM 代码中有一个完全的24位的数字转换的子程序。

在汇编程序中这个子程序是以操作码no2lcd出现的。在X-累加器的数值被转换然后被写入LCD内存的低48位(lcd_segment[39:0]带有符号的原始数值被写回到X-累加器然后可以用来设置在显示器上的符号。小数点的位置在Y-累加器中被显示。小数点前面的0不会显示。LCD驱动忽略了上面在2xmultiplex中的上两个数字和在3xmultiplex中的上面的数字。在3x和4xmultiplex中的特殊字符将不能被改变(lcd_segment[55:48])。在2xmultiplex中显示的小数点可也用于特殊字符。在这个情况下他们必须在转换之后重新存储。

注：

必须要分别通知LCD驱动关于在lcd 寄存器13-15 中的新数据。这个是通过操作码newlcd来完成。

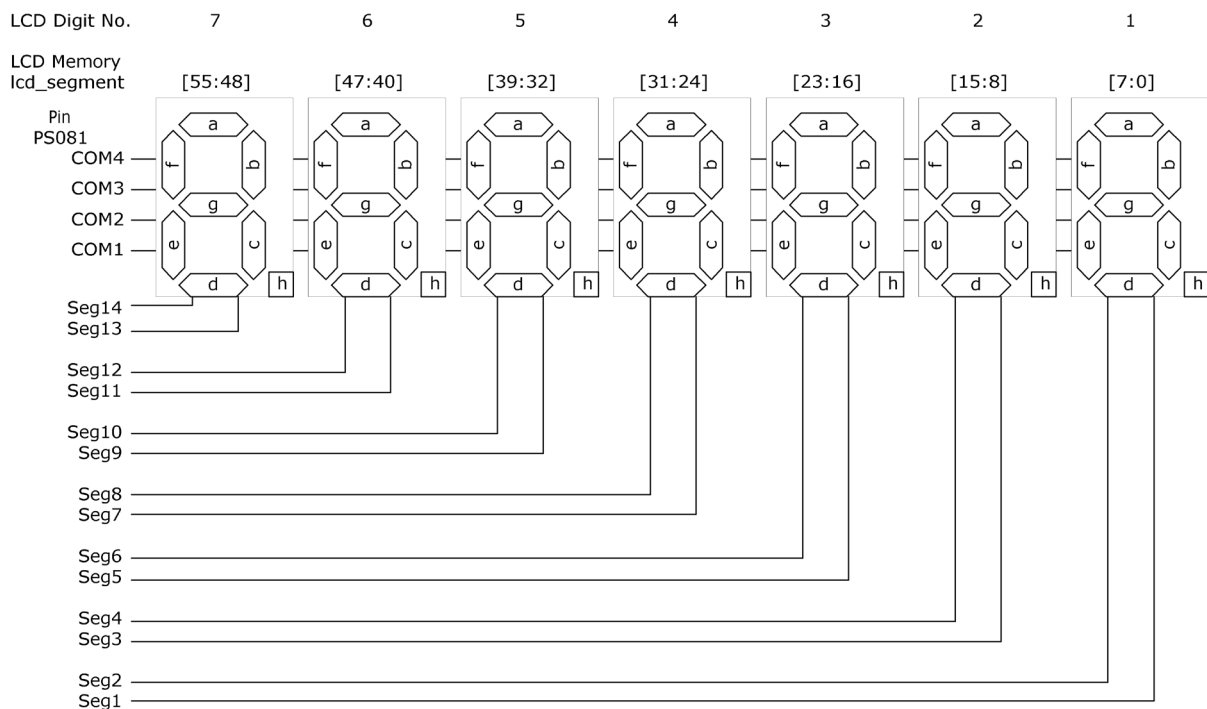
样例程序：

```
ramadr 20                ; HB0结果
move    x , r             ; 加载结果到X-累加器
move    y , 2             ; 加载小数点的位置到Y-累加器
no2lcd   ; 转换到7段码显示格式
newlcd   ; 刷新LCD
clrwdt   ; 重置看门狗定时
stop     ; 停止uC
```

4.2.5 连接原理图

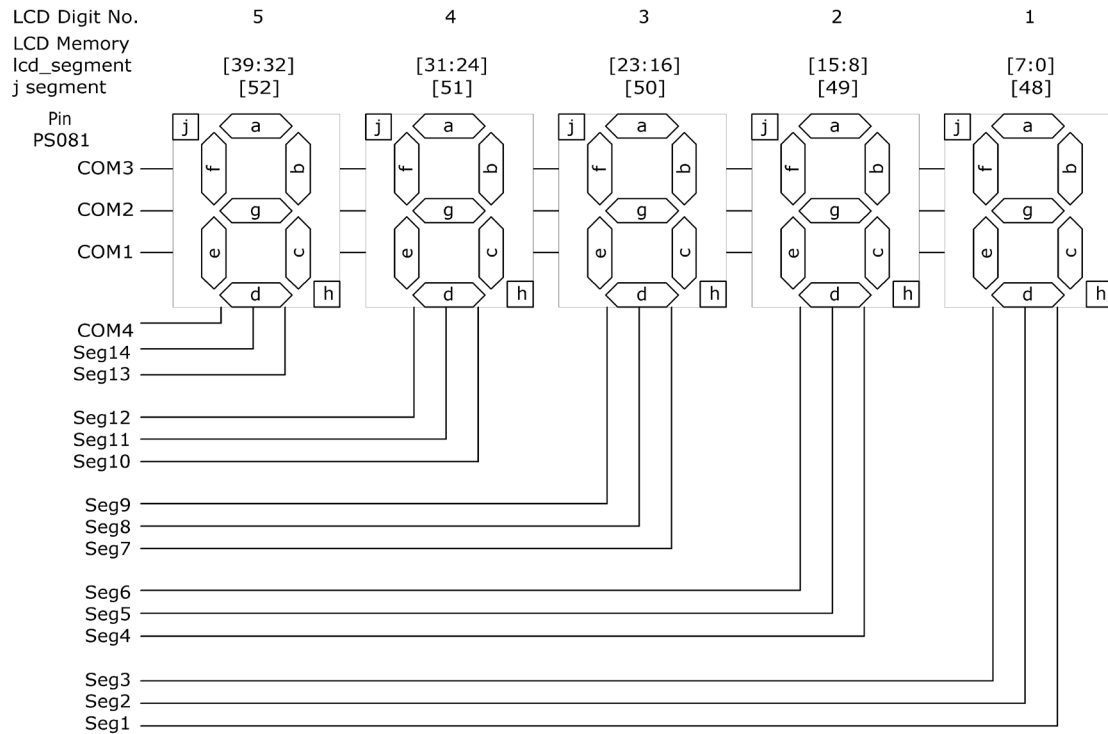
4.2.5.1 4-MUX (1/4 duty)

图 4.6 4-MUX (1/4 duty)



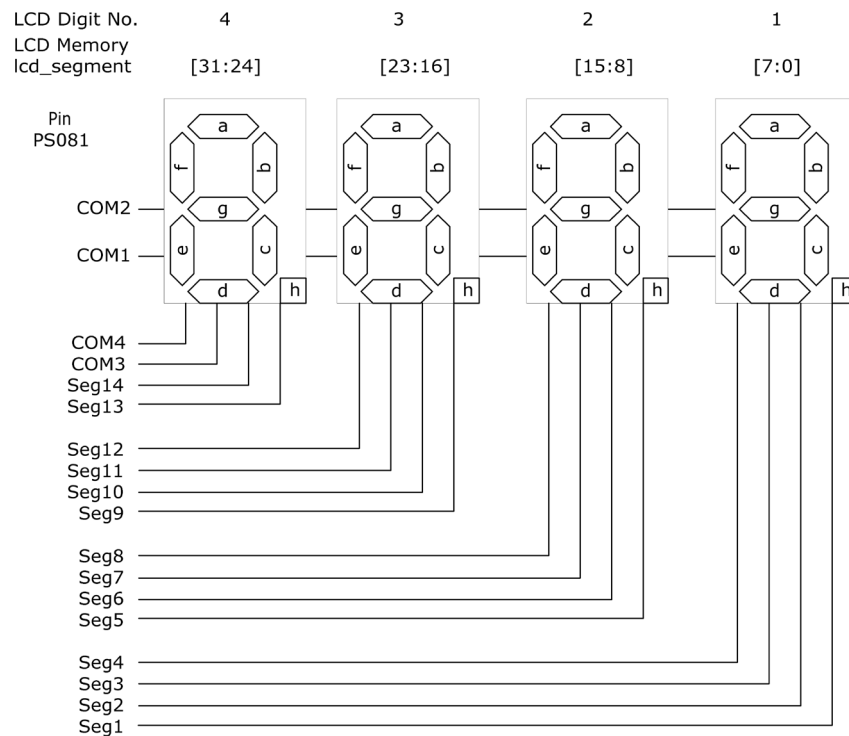
4.2.5.2 3-MUX (1/3 duty)

图 4.7 3-MUX (1/3 duty)



4.2.5.3 2-MUX (1/2 duty)

图 4.8 2-MUX (1/2 duty)



4.2.6 设置各段的位置

每一个lcd_segment控制位的段都可以任意连接到公共驱动线和段码驱动线上。这一点提供了高的驱动灵活性而且允许连接现有的不同的LCD。

限制:

段码线和实际的公共线3, 4必须要连接在正确的数字上。在一个数字里面的顺序可以是随意的。否则，no2lcd 将会搞混所有数字。

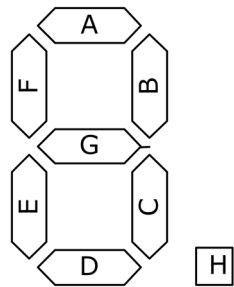
在寄存器 lcd_segment 中程序设置了被显示的各段。在 lcd_pos 中则定义了 lcd_seg-ment 中具体哪一位对应8个目标段的位置。

例: 4X Mux显示

PS081的默认连接:

	最左面的数字							最右面的数字	
	Seg	Seg	Seg	Seg	...	...	...	Seg	Seg
	14	13	12	11	...	...	...	2	1
Com1	lcd_pos [14...12]	lcd_pos [2...0]	lcd_pos [14...12]	lcd_pos [2...0]	...	...	...	lcd_pos [14...12]	lcd_pos [2...0]
Com2	lcd_pos [17...15]	lcd_pos [5...3]	lcd_pos [17...15]	lcd_pos [5...3]	...	...	...	lcd_pos [17...15]	lcd_pos [5...3]
Com3	lcd_pos [20...18]	lcd_pos [8...6]	lcd_pos [20...18]	lcd_pos [8...6]	...	...	...	lcd_pos [20...18]	lcd_pos [8...6]
Com4	lcd_pos [23...21]	lcd_pos [11...9]	lcd_pos [23...21]	lcd_pos [11...9]	...	...	...	lcd_pos [23...21]	lcd_pos [11...9]

图 4.9 4 mux-digit



对应的段	lcd_pos[] =
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	7
i	8

例 1:

LCD 说明

	Seg 2, 4, 6...	Seg 1, 3, 5...
Com1	e	a
Com2	f	b
Com3	g	c
Com4	h	d

那么

lcd_pos 值

Seg 2, 4, 6...	Seg 1, 3, 5...
lcd_pos [14...12] = 4	lcd_pos [2...0] = 0
lcd_pos [17...15] = 5	lcd_pos [5...3] = 1
lcd_pos [20...18] = 6	lcd_pos [8...6] = 2
lcd_pos [23...21] = 7	lcd_pos [11...9] = 3

例 2:

LCD 说明

	Seg 2, 4, 6...	Seg 1, 3, 5...
Com1	g	a
Com2	b	f
Com3	e	e
Com4	h	d

那么

lcd_pos 值

Seg 2, 4, 6...	Seg 1, 3, 5...
lcd_pos [14...12] = 6	lcd_pos [2...0] = 0
lcd_pos [17...15] = 1	lcd_pos [5...3] = 5
lcd_pos [20...18] = 2	lcd_pos [8...6] = 4
lcd_pos [23...21] = 7	lcd_pos [11...9] = 3

lcd_pos[23...0] = 111110101100011010001000
= 0xFAC688

lcd_pos[23...0] = 111010001110011100101000
= 0xE8E728

例：3X Mux 显示

PS081的默认连接:

	最左面的数字						最右面的数字		
	Com	Seg	Seg	...	...	...	Seg	Seg	Seg
	4	14	13	...	...	...	3	2	1
Com1	lcd_pos [20...18]	lcd_pos [11...9]	lcd_pos [2...0]	...	...	...	lcd_pos [20...18]	lcd_pos [11...9]	lcd_pos [2...0]
Com2	lcd_pos [23...21]	lcd_pos [14...12]	lcd_pos [5...3]	...	...	...	lcd_pos [23...21]	lcd_pos [14...12]	lcd_pos [5...3]
Com3	**	lcd_pos [17...15]	lcd_pos [8...6]	...	...	...	**	lcd_pos [17...15]	lcd_pos [8...6]

例1:

LCD 说明

	Seg 3, 6, 9...	Seg 2, 5, 8...	Seg 1, 4, 7...
Com1	g	d	a
Com2	h	e	b
Com3	i	f	c

那么

lcd_pos 数值

Seg 3, 6, 9...	Seg 2, 5, 8...	Seg 1, 4, 7...
lcd_pos [20...18] = 6	lcd_pos [11...9] = 3	lcd_pos [2...0] = 0
lcd_pos [23...21] = 7	lcd_pos [14...12] = 4	lcd_pos [5...3] = 1
**	lcd_pos [17...15] = 5	lcd_pos [8...6] = 2

lcd_pos[23...0] = 111110101100011010001000
= 0xFAC688

** 特殊符号是由LCD设计所确定的，并不能被重新布置。注意这些部分连接到Com3和Seg3, 6, 9...

例2:

如果

	Seg 3, 6, 9...	Seg 2, 5, 8...	Seg 1, 4, 7...
Com1	h	e	d
Com2	f	g	c
Com3	i	a	b

那么

Seg 3, 6, 9...	Seg 2, 5, 8...	Seg 1, 4, 7...
lcd_pos [20...18] = 7	lcd_pos [11...9] = 4	lcd_pos [2...0] = 3
lcd_pos [23...21] = 5	lcd_pos [14...12] = 6	lcd_pos [5...3] = 2
**	lcd_pos [17...15] = 0	lcd_pos [8...6] = 1

lcd_pos[23...0] = 101111000110100001010011
= 0xBC6853

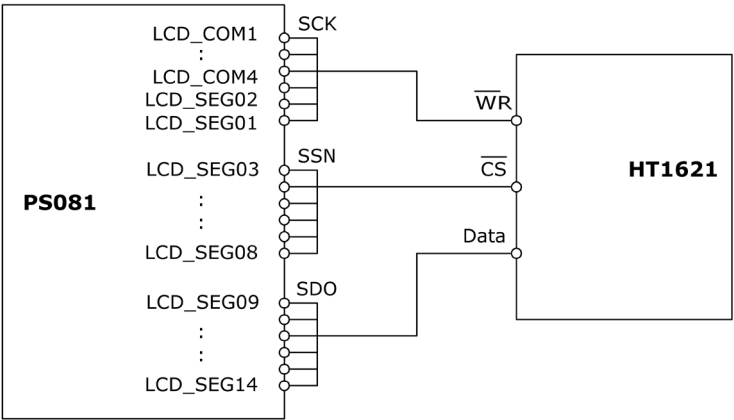
** 特殊符号是由LCD设计所确定的，并不能被重新布置。注意这些部分连接到Com3和Seg3, 6, 9...

4.3 对于外部LCD驱动的支持

PS081提供了控制外部LCD驱动的可能。因此我们在PS081中集成了一个简单的SPI主模式，尤其针对Holtek HT1621和FM upd7225设备非常有效。
使用LCD驱动的端口作为驱动外部LCD驱动SPI接口。为了提供足够的驱动电压应用了6个并行的接口。

外部LCD驱动连接电路:

图 4.10 外部LCD驱动器



配置:

- lcd_spi_ena
- 1 = 开启LCD引脚的SPI主模式
- lcd_duty
- 2 = 在LCD作为SPI的主驱动时，必须要设置一个duty模式。
(1 = 1/2 duty, 2 = 1/3 duty, 3 = 1/4 duty)
- lcd_directdrive
- 1 = VCC直接提供给LCD电压
- lcd_standby
- 0 = LCD电压产生可以被关闭

lcd_spi_out[2:0] 设置LCD引脚为SPI输出，6个平行的引脚
 xx1 开启 LCD_COM1...LCD_COM4, LCD_SEG1, LCD_SEG2 作为 SCK信号线
 x1x 开启LCD_SEG3...LCD_SEG8 作为 SSN信号线
 1xx 开启LCD_SEG9...LCD_SEG14 作为 SD0信号线
 spi_delay[2:0] 为LCD-SPI主模式接口设置时序
 0 = 快速(大约500ns周期)
 :
 7 = 慢速(大约10μs周期)

编程：

在no2lcd 操作码后，寄存器61和62根据标准配置保持了LCD位素的数据。然后根据lcd_pos在LCD中完成重新布置。在操作码new_lcd之后重新分布的数据可以在RAM地址中读出：

reg_lcd_pix_sort61= 64+32+2 = 98

reg_lcd_pix_sort62= 64+32+3 = 99

在指令new_lcd之后等几个周期是有必要的。

这些是支持LCD SPI主机控制的新操作码：

ssnPulse 将会在 „SSN “线上产生一个正脉冲 (lines LCD_SEG3...LCD_SEG8)
 ssnSet 设置 „SSN “为高
 spi2lcd 这个操作码发送了累加器Y的内容到SPI端口。第2个参数则定义了多少位将会被传输。这个功能主要是应用在当需要通过SPI外接一个LCD驱动器的时候所设。

注：

- no2lcd操作码规定了输出给外部LCD的数字格式。它总是与newlcd操作码结合起来使用。
- 这可能需要你根据自己的LCD来修改寄存器12中的lcd_pos。
- 请注意累加器-X、Y和Z因为操作码no2lcd和spi2lcd被使用和修改。
- 不可能同时连接以下两者内、外部LCD。一次只能驱动一种模式。

样例：

;初始化

jsub init_holtek

;测量

ramadr 20 ;HB0

move x, r

no2lcd x, 1

newlcd

jsub driver_4

;其他的函数

请参照附录、子程序说明driver_4和init_holte。一般来说，单片机驱动需要完成赋初值然后newlcd操作码通过程序driver_4(MUX4 LCD)发送给LCD。

4.4 I/O 管脚

PS081有6个I/O管脚

0 - SPI_DO_I00	串行数据输出(SD0)或者多功能 I/O
1 - SPI_DI_I01	串行数据输入(SD0)或者多功能 I/O
2 - SPI_CLK_I02	串行时钟(SD0)或者多功能I/O
3 - MULT_I03	在惠斯通应用中为多路模拟输出选择，阻止输出或者 多功能I/O
4 - MULT_I04	多功能I/O
5 - MULT_I05	多功能I/O

芯片在stand-alone模式的情况下，这些管脚可以被配置为带有上拉或下拉电阻的输入或输出管脚（SPI接口不被使用，SPI_ENA=0）。只有在不是惠斯通模式下管脚MULT_103才可以用来作为输入或者输出。如果所有这些管脚都没有配置为输出, 那么输入管脚的个数最多可以增加21个, 如后面章节所述。

另外，管脚24 SPI_CSN_RST在SPI接口不使用的情况下可以用于复位输入（SPI_ENA = 0）。复位是高平有效。

4.4.1 配置

SPI_DO_I00	Configreg_11, bit 16, 17 io_en_3_sdo
SPI_DI_I01	Configreg_11, bit 18, 19 io_en_3_sdi
SPI_CLK_I02	Configreg_11, bit 20, 21 io_en_3_sck
MULT_I03	Configreg_11, bit 22, 23 io_en_3_mio
MULT_I04	Configreg_17, bit 2, 3 en_io4
MULT_I05	Configreg_17, bit 4, 5 en_io5

4.4.2 端口定义

00 = 输出
01 = 输入带上拉
10 = 输入带下来
11 = 输入

MULT_I04 & 05 根据默认设置为输入。MULT_I04设置为01因为在PS081中这一管脚是用来连接VCC的。在这种情况下，管脚与旧版本芯片的兼容性被保持。

4.4.3 输出-写操作

输出设置是在配置寄存器1和17中:

SPI_DO_I00	Configreg_00, bit 10	io_a[0]
SPI_DI_I01	Configreg_00, bit 11	io_a[1]
SPI_CLK_I02	Configreg_00, bit 12	io_a[2]
MULT_I03	Configreg_00, bit 13	io_a[3]
MULT_I04	Configreg_17, bit 6	io_a[0]
MULT_I05	Configreg_17, bit 7	io_a[1]

4.4.4 增加I/O个数

通过外接的二极管, 输入管脚能实现的最大数值是21个, 只要没有管脚用作输出。这些管脚与您连接的管脚数有关, 那么最大的输入个数将会有所不同. 下面的表格和图片将会显示了连接输入的情况. 请注意在惠斯通模式下, 将会自动减少一个I/O管脚, 因为I/O3将会被用作惠斯通模式的外部开关选择.

表 4.3 输入的个数

管脚数:	输入的个数:
6	21
5	15
4	10
3	6

2个输入一起可以构成另一个输入。这两条线越过二极管电路进行连接。请看下图所示最大的结合输入个数（21）。增加输入个数仅在您需要多于4个输入的时候才有意义, 否则您可以直接应用 Mult_IO 0 到 Mult IO 3管脚.

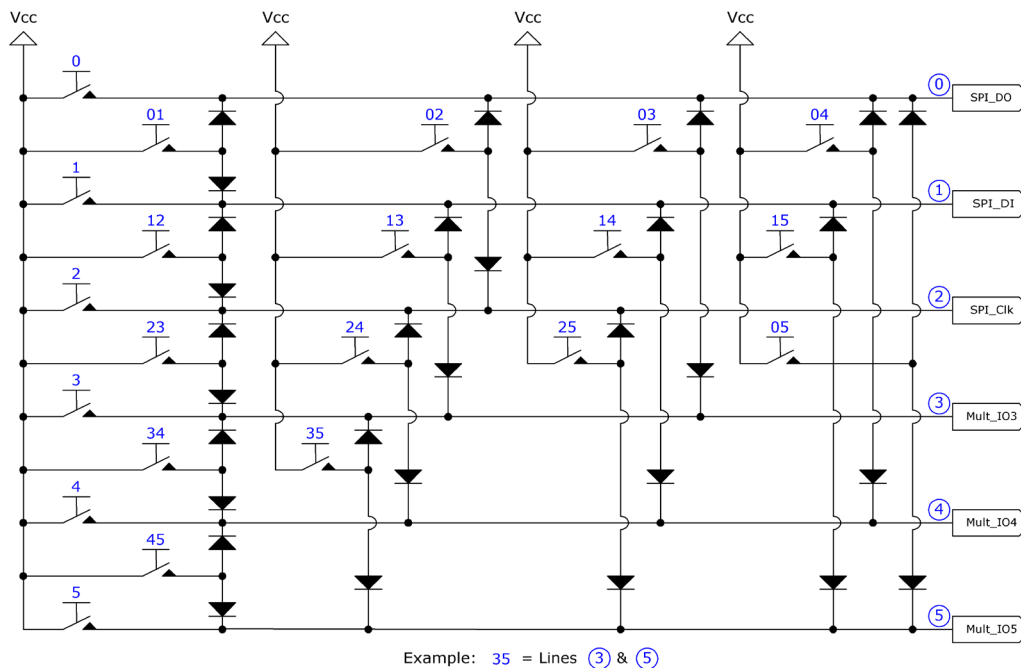
对于同时按下按钮的识别是与时序有关系的, 因此状态寄存器RAM地址26-28被配置的具有以下特点:

- 同时按下的按钮只有在单独的按钮之前没有被按下的情况才会有效
- 如果同时按下的按钮被确定, 那么其中一个按钮的信息将被忽略
- 一次只能确认一对同步按下按钮, 其他的按钮将被忽略。

可以按下多个不同的单个按钮, 但是需要按顺序, 而非同时按下

请注意如果您连接了多个输入的话, 对于每一个管脚的监测请通过状态寄存器26 到 28来进行. 您不能够通过 register 22 来判断!

图 4.11 最大组合IO输入个数



4.4.5 输入 - 读

4.4.5.1 4 个输入或者更少 / 输入和输出混合的情况

输入I00到I03的中断标志将会在状态寄存器, 地址22中被显示. 另外会对于上升沿, 下降沿, 或者按钮按下动作也有显示:

状态信息 RAM 地址 22 *) (与以前的 PS08相同)

Status[23]= flg_io3_mio

Status[22]= flg_io2_sck

Status[21]= flg_io1_sdi

Status[20]= flg_io0_sdo

Status[07]= flg_io3_mio_r 上升沿

Status[06]= flg_io2_sck_r 上升沿

Status[05]= flg_io1_sdi_r 上升沿

Status[04]= flg_io0_sdo_r 上升沿

Status[03]= flg_io3_mio_f 下降沿

Status[02]= flg_io2_sck_f 下降沿

Status[01]= flg_io1_sdi_f 下降沿

Status[00]= flg_io0_sdo_f 下降沿

4.4.5.2 多于 4 个输入的情况, 没有输出

输入的状态可以通过在状态寄存器 RAM address 26 到 28 当中进行监测.

Status_ R, address 27: 上升沿状态

Status_ F, address 26: 下降沿状态

Status_ P, address 28: 按钮按下状态

位:	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
输入:	05	15	04	25	14	03	35	24	13	02	45	34	23	12	01	5	4	3	2	1	0

如果两个输入在同一时刻同时开启, 那么PS081将会将这个动作看作为额外的按钮动作. 比如, 如果按钮 MULT_IO3 和 MULT_IO4同时刻被置高, 那么位[14] 将会在 Status_R 和 Status_P中被置位. 这将使输入按钮的个数最多上升到21个 (见下图).

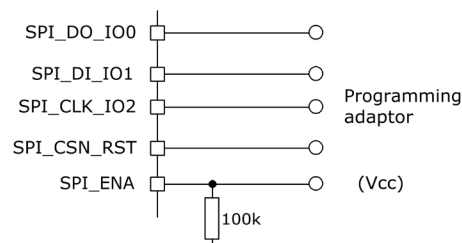
4.5 SPI 接口

4.5.1 接口

SPI接口用于写入EEPROM里面程序, 配置数据, 校准数据. 它还可以用于当PS081作为纯的转换器时与外部单片机进行通信. 在这个情况下下拉电阻将不必使用.

将SPI_ENA置高将会开启SPI接口, 那么所属管脚将作为SPI接口的通信管脚不再作为普通I/O口用. 在每次操作码发送之前必须要发送一个正脉冲给CSN线.

图 4.12 SPI 接口



4.5.2 SPI 时序

这里我们仅描述了PS081作为纯时间数字转换器时的SPI与外部单片机通信的操作. PS081仅支持4个可能的通信模式中的一个模式:

时钟相位时钟相位 = 1, 时钟极性时钟极性 = 0

数据是在时钟的第二个跳变沿被传输, 时钟是从低电平开始。

图 4.13 SPI时序

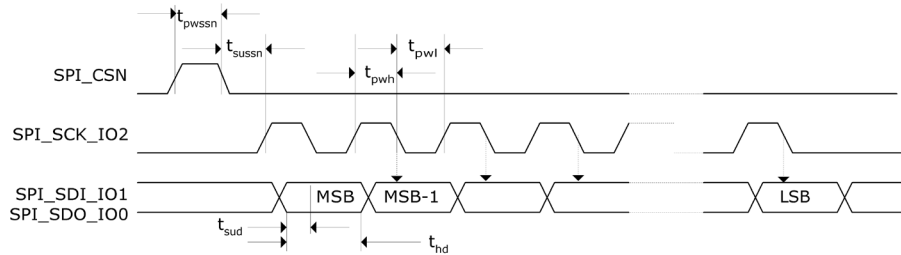


表 4.4 串行接口通信时序

时间:	描述:	t_{min} [ns]
tpwssn	脉宽 SSN	500
tsussn	启动时间 SSN / SCK	500
tpwh	脉宽 SCK 高平	500
tpwl	脉宽 SCK 低平	500
tsud	数据准备时间	30
thd	数据保持时间	30

Tpwh和tpwl两者一起定义了SPI接口时钟频率。因此1us相当于SPI以1MHz的速度传输。通过SPI发送一个复位之后，有必要在发送下一个操作码之前等待200us。如果是自动配置，有必要等待1ms。在通过SPI写入RAM之后等待10us。

4.5.3 SPI 指令

4.5.3.1 RAM 操作

RAM Write	= b00000000	= h00	
RAM Read	= b01000000	= h40	
New_LCD	= b01000110	= h46	
Power reset	= b11110000	= hF0	
Init reset	= b11000000	= hC0	
Start_new_cycle	= b11001100	= hCC	(持续测量)
Start_TDC_cycle	= b11001110	= hCE	(单一转换模式)
watch_dog_off	= b10011110	= h9E	
watch_dog_on	= b10011111	= h9F	

4.5.3.2 EEPROM 操作:

EEprom_bgap_off	= b10000110	= h86
EEprom_bgap_on	= b10000111	= h87
EEprom_enable_off	= b10010000	= h90
EEprom_enable_on	= b10010001	= h91

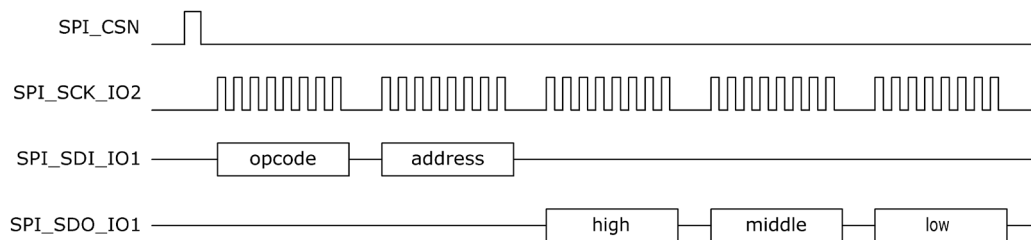
EEprom_read	= b10100000	= hA0	(读保护)
EEprom_write	= b10100001	= hA1	
EEprom_bwrite	= b10100011	= hA3	
EEprom_berase	= b10100100	= hA4	
userEEprom_read	= b10100101	= hA5	地址 2000-2047

在对EEPROM进行读写之前要开启带隙来准许进行操作。

(发送 EEprom_bgap_on, EEprom_enable_on)

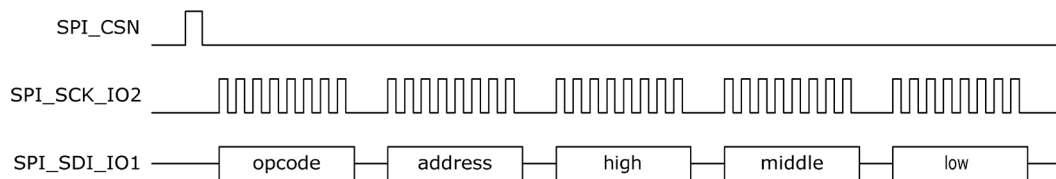
RAM 读操作

图 4.14 RAM 读操作



RAM 写操作

图 4.15 RAM 写操作

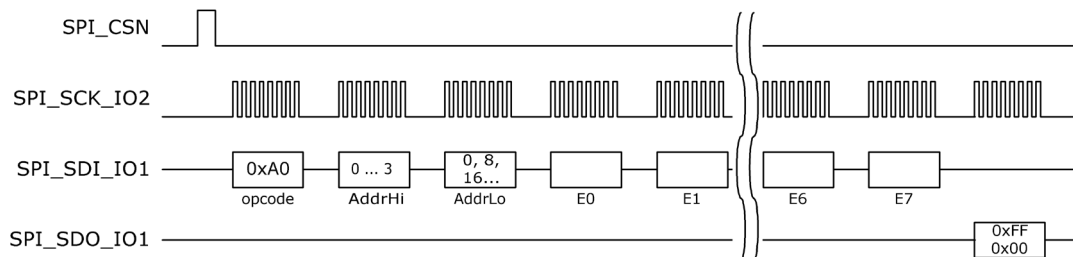


EEPROM 读操作/读保护

PS081读 EEPROM是受到保护的，当在没有授权情况下内容是不能被读出的。仅仅当已知的内容与EEPROM中的内容完全相同的时候，才能够进行操作。读地址0到255的同时将会检查两个EEPROM的16位字。因此命令 EEProm_read 是跟在 EEPROM1 高字节，EEPROM1 低字节，EEPROM2 高字节，EEPROM2 字节之后的，那么芯片将会用所传输的数据与EEPROM的内容向比较。结果将会在1ms后在 SDO 端口有效。0xFF代表正确，0x00代表错误数据。

没有被授权的人如果想进行读操作必须要测试所有的组合。这将会持续 $2^{64} \text{ bits} * 10 \text{ ns} = 5849 \text{ 年}$ ，仅是来读到所有内容。

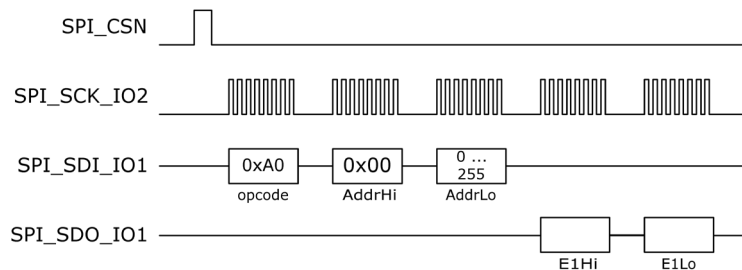
图 4.16 读操作/读保护



用户-EEPROM 单元的读操作

EEPROM的高48字节，地址从2000到2047，被用作用户EEPROM单元。可以通过发送操作码 `userEEPROM_read` 来读取其中内容。`userEEPROM_read` 将会读回16位字。

图 4.17 用户EEPROM读操作

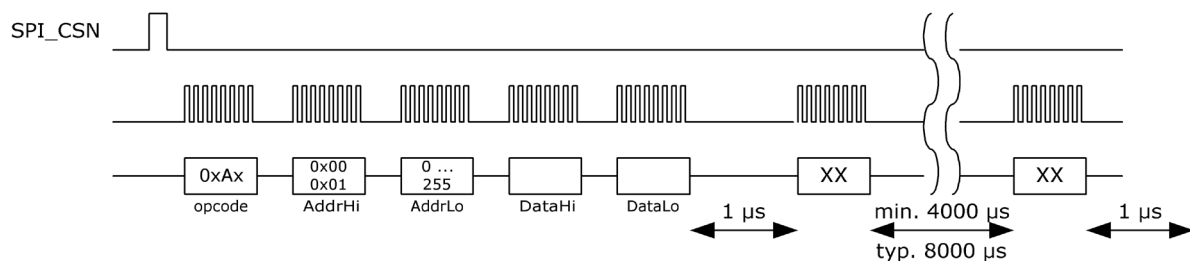


EEPROM 写操作

EEPROM被分为2个512 bytes的模块，或者2个256字符的模块。那么操作不同模块是根据发送地址码的最低那个位而决定的。在发送写命令和地址码后跟随发送的是16bit数据的高8位，然后再发低8位。烧写EEPROM过程是在第6个8位字写进之后开始的，那么至少在4ms后，再发送第7个8位字节写进之后烧写结束。

EEPROM单元有内容的话是不能写入字符的。因此，不可能写入一个字到0xFFFF（另一种测量方式以防止非法读取）。

图 4.18 EEPROM 的写操作



EEPROM擦除

擦除整个模块AddrHi内的地址。看起来和EEProm_write一样，但是AddrLo和数据是被忽略的。擦除整个EEPROM的指令将被发送4次（AddrHi = 0, 1, 2 & 3）。

4.5.4 外部单片机通过SPI端口控制PS081 (非-连续 配置)

在开始之前： 请先确定 EEPROM 为空.

配置芯片以及开启测量的命令顺序：

Power Reset (0xF0)

Watchdog off (0x9E)

在RAM当中对PS081进行配置：

RAM Write (0x00) + 地址 + 3x8Bit

Write RAM 地址 48..66

重要： 所有的 epr_xxx_xxx 位都要设置为零
(configreg_1, bit 0-2)

可选择： 控制读取RAM配置

RAM 读 (0x40) RAM 地址 48..64

寄存器的配置完成.

Init Reset (0xC0)

Start New Cycle (0xCC)

查询 / 中断 SPI_DO: 新的测量结果的给出是以 SPI_DO 1 -> 0为标志的。

可以通过单片机来查询这个管脚或者直接接到单片机的中断管脚。

当 SPI_DO 从 1 变到 0的时候，切换SPI_CS的电平进行变化为 0 -> 1 -> 0

(通过这种方式 SPI_DO 将会被开启作为SPI通信应用)：

在RAM 地址 0 读取 HB0 的结果 (RAM Read, 0x40)

注释:

- 请一定确定在进行通信开始前首先删除EEPROM的内容。
- SPI_ENA 必须为高平 HIGH. 这个动作将会选择SPI-从机模式另外开启SPI通信端口。
- 当测量开始进行之后，新的数据的完成将会由 SPI_DO的状态显示出来。将这个管脚连接到您单片机的输入管脚，然后对这个管脚进行查询，或者直接接到单片机的中断管脚。
- 当中断被触发之后，请切换 SPI_CSN 管脚的电平来将SPI_DO管脚从中断源切换到通信模式当中(参见下图)
- 在括号当中的Hex数字为SPI的操作码。请参见PS081手册的章节4.5 SPI-接口来获得详细的描述。
-

- 还有一种方法可以将配置文档持续的存放在EEPROM当中，然后仅在上电复位的时候激活这个配置。如果您希望应用这个功能请向acam公司工程师咨询。
- 删除EEPROM，如果需要的话，可以通过如下方式实现：

Power reset (0xF0)

删除 EEPROM: Bgap on (0x87)

EEPROM enable on (0x91)

EEPROM block erase 0 (0xA4, AddHi = 0)

EEPROM block erase 1 (0xA4, AddHi = 1)

EEPROM block erase 2 (0xA4, AddHi = 2)

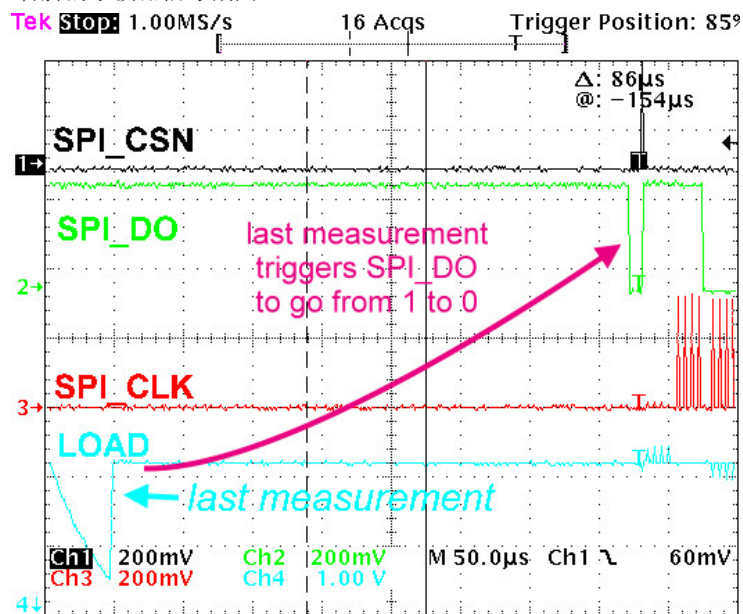
EEPROM block erase 3 (0xA4, AddHi = 3)

Bgap off (0x86)

EEPROM enable off (0x90)

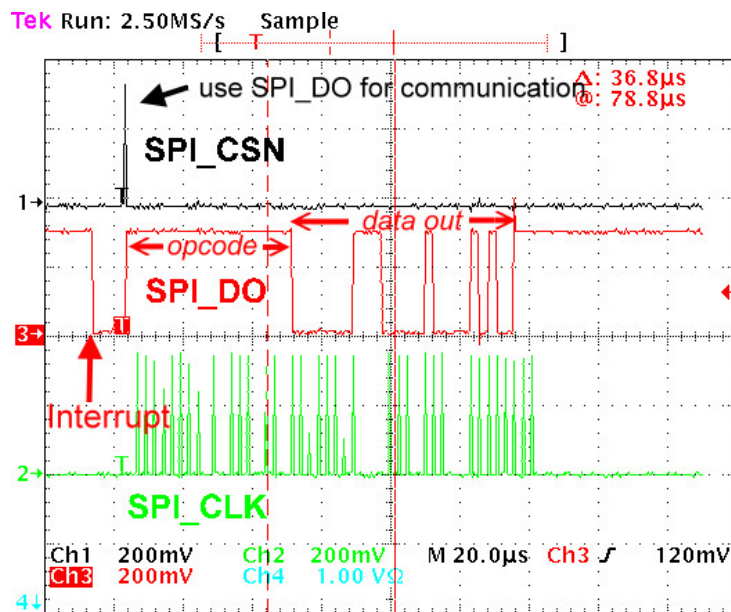
推荐通过RAM address 0 来读取HB0的测量结果。如果从RAM address 20 (同样也是HB0 结果) 读取结果的话, 那么有可能会造成地址指针的冲突, 因此请从RAM address 0来读取结果。请注意, HB0的结果在EEPROM中没有任何程序的情况下(作为纯转换器应用的时候)是自动被拷贝存放在RAM address 0 中的。**如果您的EEPROM中有额外的程序的话(比如数据预处理等), 那么您需要手动将HB0的结果从address 20 拷贝到 address 0 当中!**

图 4-19: Load管脚和SPI管脚的示波器信号截图



在图4.19 当中您可以看到SPI D0 在最后测量放电循环结束后将会给出一个中断信号。

图. 4.20: 在中断后读取测量结果



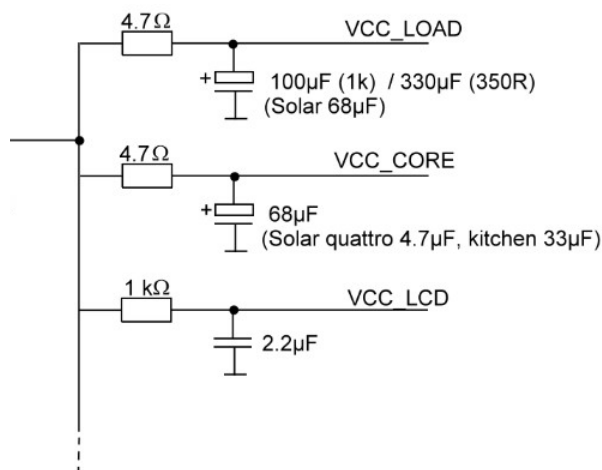
这个示波器的截图显示了SPI_这个示波器截图同样显示了 SPI_DO中断从 1 → 0，之后外部单片机的SPI_CSN管脚的响应情况。在SPI_CSN上的短暂得脉冲将会将 SPI_DO 切换到通信模式当中。然后发送操作码到PS081（没有在图中显示，这个发送是在SPI_DI线给出）在 2 bytes之后测量结果（3 bytes）将会通过 SPI_DO传输给单片机（在图中标记为 ,data out ‘）。

4.6 电源电压

为了获得好的测量质量必须要严格按照下面的规则来设计电源电压。

芯片有几个供电区域VCC_LOAD、VCC_CORE和VCC_LCD。通过低通滤波器对电压进行去藕是有必要的。另外这些管脚需要足够的滤波电容而且放置的要离芯片管脚近。

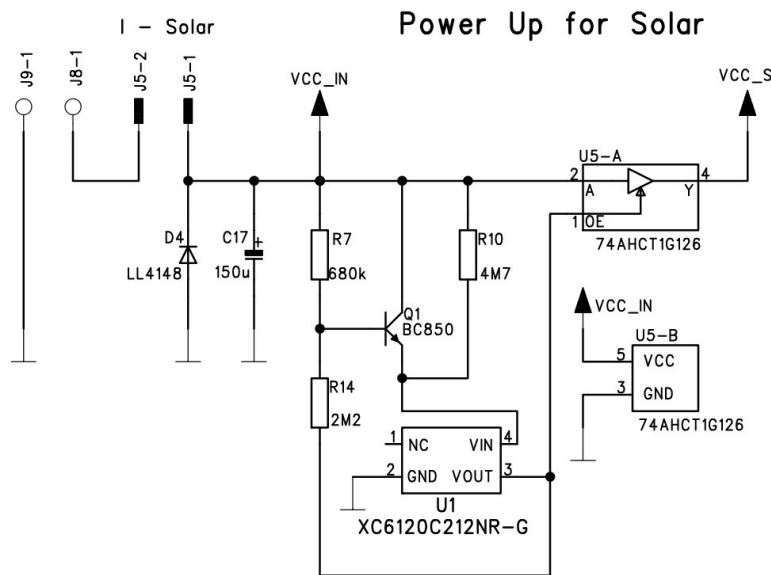
图 4.21 电源电压



在太阳能应用并且不带任何另外的电源时，必须要使用上电电路。这个电路给出了一个非常好的上电特性，尤其当秤从黑暗到有光的这个过程的情况下。太阳能电池板将会在很暗的情况下给出大约几个微安的电流，因此必须要进行这个上电的操作。

下面的图说明了一个在太阳能厨房秤当中的上电应用电路。这个电路能够在3.6V，3uA的电流情况下上电。对于其他的太阳能应用可能需要改变元器件的数值大小。

图 4.22 太阳能应用上电电路



对于上电功能的简单描述:

在完全黑暗的状态下，所有的电容将会被完全放电，那么U5的输出是搞阻抗的状态。U4的输入为零。PS081这个时候是没有电压供电的。

如果光源被开启，那么太阳能电池板将会给C17进行充电，然后给电压检测电路提供电源。电压检测电路上元件(R7, R14, Q1, U3)大小的选择目的主要是当在C17的电压大于3.5V的时候，U1将会开启。在这个时刻，U5的输出将会脱离高阻状态而将会与C17的电压相同。U4是通过3.5V供电然后将电压调整到2.5V给到PS081。PS081在这个时候开始工作。因为所有的在VCC_R的电容都需要充电到C17的电压，这个电压将会下降，那么C17仅会提供必须的电流。太阳能电池板对于如此大的电流脉冲来说是比较弱的。C17的电压将会下降，但是不能够降低到2.55V一下。否则U4将不能将2.5V的调节电压给到PS081。C17也是在光线很暗的情况下一个缓冲供电电容。通过选择正确的电容大小，在非常暗的情况(20 Lux)衡器可以在正常充电后工作至少1分钟。因此我们在这个厨房秤应用中推荐 C17的值为1000 μ F (最小 680 μ F)。

请一定不要将这个电容设计低于680 μ F。如果C17太小的话电路是不能够上电的。电路在C17比较大的情况下也是可以工作的。但是值越高，那么上电到正常工作的时间将会越长。

这个电路是经过细致的研究所设计，因此我们强烈推荐在同样应用中用正确的数值。如果需要进一步的信息请参考应用指南 AN022 以及设计指南 DG_Solar_POR。

4.6.1 滤波 / 推荐的 LDO

在大多数的电路当中电压是通过一个电压调节元件(LDO, low drop-out regulator)进行调节整形的。当然这个元器件本身的噪声会对测量的质量有一定的影响。因此，选择一个合适的低噪声的LDO是非常有必要的，而且有一些应用还需要低功耗类型的LDO。在这个章节当中我们给出一些对于LDO选择上的建议。

最主要的一个参数需要注意的就是'output noise' 输出噪声。噪声的具体指标一般可以在LDO的数据手册当中找到，另外会有一个针对整个频率范围给出的总的的数据，比如500 μ V RMS 或者与频率有关的单位为 μ V / $\sqrt{\text{Hz}}$ 的数据，或者一个图表的形式。这里我们需要一个低输出噪声的，这个参数有时候其变化在不同芯片会有10倍的变化。

(比如 Linear LT1761-BYP 输出噪声为20 μ V RMS (带有低通滤波) vs. TI TPS71501 则有 575 μ V RMS)

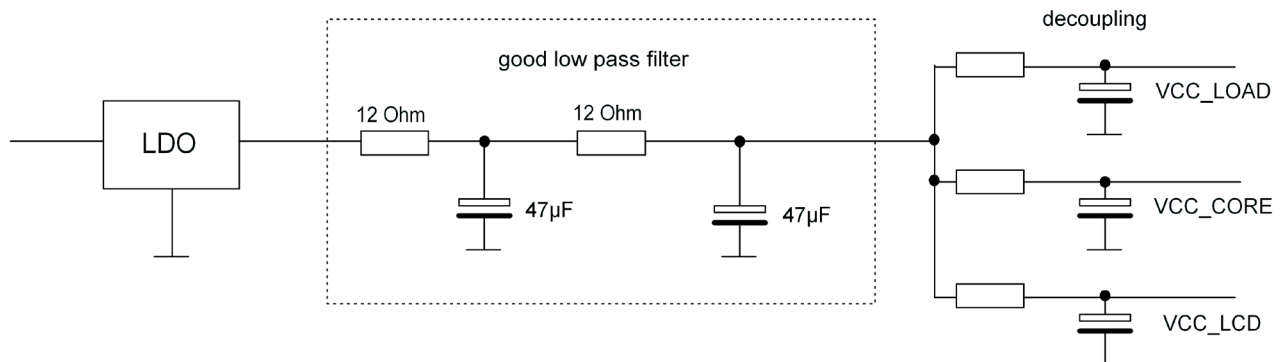
推荐：

LDO	特性	低通滤波	应用
Torex XC2206	中等噪声，低功耗	应用好的低通滤波元件	太阳能方面
Micrel MC5205	中等噪声，价格便宜	应用中等的低通滤波	低价应用
Linear LT1761-BYP	超低噪声，价格较高	用标准的低通滤波	高精度应用
TI TPS71501	高噪声	-	不推荐使用!!!

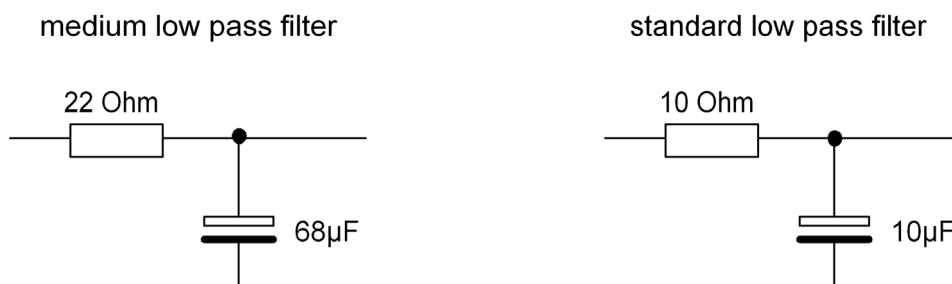
当然，这个表格仅为根据我们的实际经验列出了一些代表作为参考，还有很多其他的LDO可以应用。当然，最重要的是一定要选低功耗和低噪声的LDO。

请不要应用TI TPS71501 LDO，我们对于这个电压元件看到非常多的噪声问题！

一个额外的低通滤波可以进一步降低进入到芯片的噪声。在给电源去藕之前可以应用不同种类的低通滤波：



好的低通滤波可以降低LDO的噪声，因此可以与电源去藕的电容一同使用来降低PS081的测量噪声。如果从LDO本身的噪声很低(例如 Linear 或者 Micrel的一些类型) 那么可以应用一个简单的低通滤波，如下：



在acam的评估系统电路当中，太阳能四角人体秤的电路的设计已经应用于实际当中。当您应用PS081设计电路的时候，请一定按照我们的推荐来选择LDO电路，好的LDO电路会带来整体更好的测量效果。

4.6.2 电压测量

一个内部带隙电压参考用于测量电压的大小。这个测量频率为40次每秒钟。结果被存放在地址25的RAM里面，UBATT。它的计算是通过公式 $\text{电压} = 2.0 \text{ V} + 1.6 \text{ V} * \text{UBATT}/64$ 完成的。

结果可以被用为

低电池电压检测：电压水平在控制寄存器low_batt[2:0]被设置

low_batt	0	1	2	3	4	5	6	7
Level (V)	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9

在状态寄存器中的标志 flg_ub_low说明是否电压低于所设置的水平。

- 电源抑制：测量电压可以用于纠正电压变化带来的增益误差。这是通过设置控制寄存器位mult_en_ub = 1和mult_ub[7:0] 来开启的。应变测量的结果将会根据 $HB = HB / (1 + UB * [-128 \dots 127] / 2^{21})$ 被校正。
- EEPROM保护：当电压低于2.4V时自动EEPROM写操作(putep)是禁止的。这将保护EEPROM不会产生数据崩溃。

注意：

如果供电电压在2.1V以下，测量将会不正确（显示数值太高）那么测量的结果不能被采用。在1.5V系统中是不能够测量PS081的电源的。

目录	页面
5 配置寄存器	5-2
5.1 概况.....	5-2
5.2 配置参数一览表（按首字母排列）	5-3
5.3 配置寄存器列表	5-6



5 配置寄存器

PS081有18个24位的配置寄存器，对应的地址从RAM的48地址到66地址。通过设置配置寄存器能够控制整个芯片，包括应变测量和LCD控制。配置寄存器的设置是从EEPROM中镜像过来的，它由EEPROM空间中的0到47个字节配置组成（48字节配置）。

有以下几种方法可以往配置寄存器中写入数据

- 由芯片内部的微处理器写入
- 由外部处理器通过SPI接口写入
- 在上电复位过程中，把EEPROM中已经存储的配置转移到RAM中

5.1 概况

表 5.1 配置寄存器一览表

配置寄存器	RAM 地址	EEPROM 相对应字节			PS08 vs. PS081 的改变
Configreg_00	48	2	1	0	无
Configreg_01	49	5	4	3	无
Configreg_02	50	8	7	6	无
Configreg_03	51	11	10	9	无
Configreg_04	52	14	13	12	无
Configreg_05	53	17	16	15	无
Configreg_06	54	20	19	18	无
Configreg_07	55	23	22	21	无
Configreg_08	56	26	25	24	无
Configreg_09	57	29	28	27	无
Configreg_10	58	32	31	30	无
Configreg_11	59	35	34	33	无
Configreg_12	60	38	37	36	无
Configreg_13	61	在EEPROM中没有被镜像			无
Configreg_14	62	在EEPROM中没有被镜像			无
Configreg_15	63	在EEPROM中没有被镜像			无
Configreg_16	64	41	40	39	是
Configreg_17	65	44	43	42	新增
Configreg_18	66	47	46	45	新增

采用内部微处理器写入:

先把寄存器配置写入EEPROM，再镜像到RAM中。LCD段码可直接写入RAM的61-63地址，不必经过EEPROM。

采用外部微处理器写入:

不需要使用EEPROM，直接把配置写入对应的RAM地址中，并且要把Configreg_01, Bits[2:0] 中的epr_pwr_cfg, epr_pwr_prg 和epr_usr_cfg 三个参数设置为0。

5.2 配置参数一览表（按首字母排列）

表 5.2 配置参数一览表（按首字母排列）

参数	寄存器	位数	推荐值
abgl_hr	1	19...22	5
auto10k	2	0	1
avrate	2	14...23	>1
bridge	3	0, 1	
calcor	10	16...23	20
clkdivopr	16	14, 15	1
clksyc	17	0, 1	1
con_comp	11	0, 1	
cpu_speed	0	1, 2	1
cytime	2	4...13	
c10_div_lcd	17	11...15	
dis_haltp_p	0	0	0
dis_noise4	3	14	1
dis_osc_startup	0	3	1
dis_pp_cycle_mod	11	14	1
dis_startdel	11	15	0
en_avcal	1	9	0
en_fullbr_p	16	17	
en_io4	17	2, 3	2
en_io5	17	4, 5	2
en_noise_c10_lcd	16	23	1
en_wheatstone	3	21	
ena_nonlin	16	16	0
epr_pwr_cfg	1	2	
epr_pwr_prg	1	1	
epr_usr_prg	1	0	
fha_en	16	11	0
force_unused_port	11	13	0
io_a	0	10...13	
io_a	17	6, 7	
io_en_0_sdo	11	16, 17	
io_en_1_sdi	11	18, 19	
io_en_2_sck	11	20, 21	
io_en_3_mio	11	22, 23	
lcd_duty	1	15, 16	
lcd_charge	16	20, 21	
lcd_directdrive	16	19	
lcd_dis_chargem	1	11	
lcd_fastld	11	8, 9	1

参数	寄存器	位数	推荐值
lcd_freq	1	12...14	
lcd_pos	12	0...23	
lcd_pulsed	17	16...19	
lcd_r_const	11	6,7	2
lcd_r_fastld	11	3,4	1
lcd_segment	13	0...23	
lcd_segment	14	0...23	
lcd_segment	15	0...7	
lcd_spi_ena	16	22	0
lcd_spi_out	17	8...10	0
lcd_standby	11	12	
lcd_swloadlk	11	5	
lcd_vlt	11	10,11	3
low_batt	1	3...5	
messb2	1	18	1
mfake	3	2,3	2
mod_caltdc	3	23	0
mod_rspan	1	6	
mult_en_pp	1	7	
mult_en_ub	1	10	
Mult_Hb1	4	0...23	
Mult_Hb2	5	0...23	
Mult_Hb3	6	0...23	
Mult_Hb4	7	0...23	
Mult_NLK	18	0...23	0
multio_sel	16	0...3	
multihit_en	16	1	1
Mult_PP	10	0...7	
Mult_TkG	8	0...23	
Mult_Tk0	9	0...23	
Mult_Ub	10	8...15	
neg_sense	3	15	0
osz10khz_fsos	0	4...9	50
portpat	2	1	1
pptemp	2	3	1
ps_dis_phaseshift	3	11	0
ps081adjust	3	4...9	33
ps_sel_calper	3	10	0
ps_speed	16	12,13	0
rspan_by_temp	1	8	

参数	寄存器	位数	推荐值
run_la_cont	16	4	0
sel_compint	11	2	
sel_compr	0	14, 15	0
sel_qha	16	7...9	2
sel_speed_mh	16	5	0
sel_start_osz	3	17...19	
sense_discharge	16	10	1
single_conversion	2	2	
speed_talu	1	23	0
spi_delay	17	21...23	3
stop_osz	3	20	0
stretch	3	12, 13	
tdc_conv_cnt	0	16...23	
tdc_sleepmode	1	17	
use_10kHz_lcd	16	18	1

5.3 配置寄存器列表

对应的位 →

参数 →

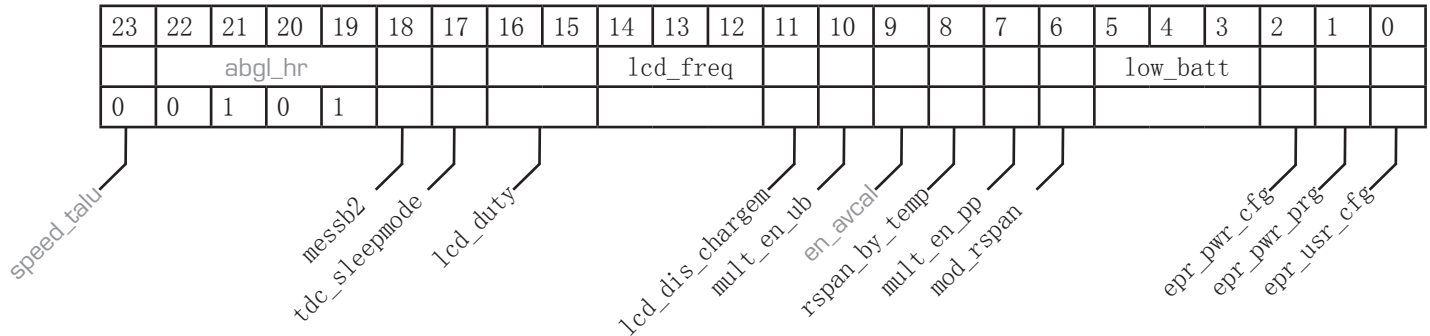
推荐 →

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
param1						param2									
						1	1	0	0	1	0	1	0	1	0

gray_labels = 灰色标签标注的位是acam内部位，需要用下面一行中给出的推荐值

Configreg_00: RAM 地址 48 EEPROM 字节 0 - 2

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
tdc_conv_cnt										io_a				osz10kHz_fsos											
														1	1	0	0	1	0	1	0	1	0		

Configreg_01: RAM 地址 49 EEPROM 字节 3 - 5


参数	推荐值	描述	设置																																																												
messb2	1	设置TDC的工作在测量范围2	1 = 开启																																																												
tdc_sleepmode	-	TDC或者应变测量都不工作的模式，其功能是在电子秤关闭的时候扫描按键是否被开启，此时avrata = 0	1 = 开启 0 = 关闭																																																												
lcd_duty	-	规定LCD占空比	0 = 关闭 1 = 1/2duty 2 = 1/3duty 3 = 1/4duty																																																												
lcd_freq		选择LCD频率（pixel的开启时间） <table><tr><th>像素</th><th>Time</th><th colspan="4">多路工作模式</th></tr><tr><td></td><td></td><td>1/4</td><td>1/3</td><td>1/2</td><td>Hz</td></tr><tr><td>0</td><td>8.0ms</td><td>15</td><td>20</td><td>31</td><td>Hz</td></tr><tr><td>1</td><td>4.8ms</td><td>26</td><td>34</td><td>52</td><td>Hz</td></tr><tr><td>2</td><td>4.0ms</td><td>31</td><td>42</td><td>62</td><td>Hz</td></tr><tr><td>3</td><td>3.2ms</td><td>30</td><td>52</td><td>78</td><td>Hz</td></tr><tr><td>4</td><td>2.4ms</td><td>52</td><td>69</td><td>104</td><td>Hz</td></tr><tr><td>5</td><td>2.0ms</td><td>62</td><td>83</td><td>125</td><td>Hz</td></tr><tr><td>6</td><td>1.6ms</td><td>78</td><td>104</td><td>176</td><td>Hz</td></tr><tr><td>7</td><td>1.2ms</td><td>104</td><td>138</td><td>208</td><td>Hz</td></tr></table>	像素	Time	多路工作模式						1/4	1/3	1/2	Hz	0	8.0ms	15	20	31	Hz	1	4.8ms	26	34	52	Hz	2	4.0ms	31	42	62	Hz	3	3.2ms	30	52	78	Hz	4	2.4ms	52	69	104	Hz	5	2.0ms	62	83	125	Hz	6	1.6ms	78	104	176	Hz	7	1.2ms	104	138	208	Hz	
像素	Time	多路工作模式																																																													
		1/4	1/3	1/2	Hz																																																										
0	8.0ms	15	20	31	Hz																																																										
1	4.8ms	26	34	52	Hz																																																										
2	4.0ms	31	42	62	Hz																																																										
3	3.2ms	30	52	78	Hz																																																										
4	2.4ms	52	69	104	Hz																																																										
5	2.0ms	62	83	125	Hz																																																										
6	1.6ms	78	104	176	Hz																																																										
7	1.2ms	104	138	208	Hz																																																										
lcd_dis_chargem		测量期间关闭LCD电荷泵																																																													
mult_en_ub		供电电压补偿使能	1 = 使能																																																												
rspan_by_temp		采用温度测量来替代Rspan来进行温度补偿	1 = 激活																																																												
mult_en_pp		增益补偿使能	1 = 使能																																																												
mod_rspan		增益补偿电阻（Rspan）内部补偿使能	1 = 使能																																																												
lowbatt		设置低电压报警和EEPROM写入的电压值	2.2 V, 2.3 V, 2.4 V to 2.9 V (2.2 V + 0.1 V * low_batt)																																																												
epr_pwr_cfg		上电复位后，EEPROM中的设置	作为转换器 := 0 作为单芯片 := 1																																																												
epr_pwr_prg		上电复位后，开始执行EEPROM地址48中存储的用户自写代码	作为转换器 := 0 作为单芯片 := 1																																																												
epr_usr_prg		测量完成后，开始执行EEPROM地址48中存储的用户自写代码	作为转换器 := 0 作为单芯片 := 1																																																												

Configreg_02: RAM 地址 50 EEPROM 字节 6 - 8

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
avrate										cytime													
																				1		1	0

pptemp
single_conversion
port_pet
auto10k

参数	推荐值	描述	设置
avrate	-	内部求平均时的采样数目	> 1
cytime	-	cycletime × 2 μs (8 * 4 MHz period, stretch = 0) 或者 × 100 μs (10 kHz period, stretch = 1)	
pptemp	1	增益误差和温度测量使能	1 = 使能
single_conversion	-	选择操作模式	0 = 连续模式 1 = 单次转换模式
auto10k	-	使用4MHz的石英晶振对10kHz晶振每0.6s进行一次自动校准	在延伸单次转换模式不能使用

Configreg_03 RAM 地址 51 EEPROM 字节 9 - 11

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														ps081adjust									
0			0				0	1		0			1	0	0	0	0	1	1	0			

mod_caldc
not used
en_wheatstone
stop_osz
sel_start_osz
not used
neg_sense
dis_noise4
stretch
ps_dis_phaseshift
ps_sel_calper

mfake
bridge

参数	推荐值	描述	设置
en_wheatstone	-	惠斯通电桥模式使能	1 = 使能
stop_osz	-	通过命令关闭晶振（例如如果在Auto On之后没有中断信号产生）	1 = 激活
sel_start_osz	-	设置在4MHz晶振开启后多久去进行测量	0 = 关 1 = 连续开启 2 = 100 μs 3 = 200 μs 4 = 300 μs 5 = 400 μs 6 & 7 无意义

参数	推荐值	描述	设置
stretch	–	选择延伸模式	0 = 关 1 = 单次转换测量 2 = 2xR (halfbridge), 200 μ s 延迟 3 = 2xR (halfbridge) 300 μ s 延迟
mfake	10	设置伪测量次数	
bridge	–	设置被测半桥数目	0 = 一个半桥 (不合理) 1 = 2 个半桥 2 = 无意义 3 = 4 个半桥

Configreg_04 RAM 地址 52 EEPROM 字节 12 – 14

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb1																							

参数	推荐值	描述	设置
Mult_Hb1	–	HB1 乘法因数 $h1 := h1 * [-2^{23} \text{ to } 2^{23-1}] / 2^{20}$	

Configreg_05 RAM 地址 53 EEPROM 字节 15 – 17

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb2																							

参数	推荐值	描述	设置
Mult_Hb2	–	HB2 乘法因数 $h1 := h1 * [-2^{23} \text{ to } 2^{23-1}] / 2^{20}$	

Configreg_06 RAM 地址 54 EEPROM 字节 18 – 20

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb3																							

参数	推荐值	描述	设置
Mult_Hb3	–	HB3 乘法因数 $h1 := h1 * [-2^{23} \text{ to } 2^{23-1}] / 2^{20}$	

Configreg_07 RAM 地址 55 EEPROM 字节 21 – 23

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Hb4																							

参数	推荐值	描述
Mult_Hb4	–	HB4 乘法因数 $h1 := h1 * [-2^{23} \text{ to } 2^{23-1}] / 2^{20}$

Configreg_08 RAM 地址 56 EEPROM 字节 24 – 26

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_TkG																							

参数	推荐值	描述
Mult_TkG	–	Rspan补偿的乘法因数 $Rs := Rs * [-2^{23} \text{ to } 2^{23-1}] / 2^{20}$

Configreg_09 RAM 地址 57 EEPROM 字节 27 – 29

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_Tk0																							

参数	推荐值	描述
Mult_Tk0	–	Rspan偏移值，可直接减去

Configreg_10 RAM 地址 58 EEPROM 字节 30 – 32

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
calcor								Mult_Ub								Mult_PP							
0	0	0	1	0	1	0	0																

参数	推荐值	描述
Mult_Ub	–	通过电压测量确定的增益补偿的乘法因数 $hb := hb / (1 + ub * [-128 \text{ to } 127] / 2^{21})$
Mult_PP	–	增益补偿的乘法因数 $g := g * [0 \text{ to } 255] / 2^7$

RAM 地址 59 EEPROM 字节 33 - 35

io_en_3_mio
io_en_2_sck
io_en_1_sdi
io_en_1_sdo
dis_startdel
dis_pp_cycle_mod
force_unused_port
lcd_standby
lcd_vlt
lcd_fastld
lcd_r_const
lcd_swloadlk
lcd_r_fastld
sel_compint
con_comp

参数	推荐值	描述	设置
io_en_0_sdo	—	端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
io_en_1_sdi		端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
io_en_2_sck		端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
io_en_3_mio		端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
lcd_standby		LCD 状态	0 = LCD 激活 1 = LCD 备用电源供电
lcd_vlt		LCD 供电电压	0 = 2V 1 = 2.5V 2 = 3V 3 = 2V （无电荷泵）
lcd_fastld		设置在使用低阻抗分压器时fast-load周期（10ms）的个数	
lcd_r_const		规定LCD分压器的交叉阻抗	0 = 30 k Ω 1 = 50 k Ω 2 = 200 k Ω 3 = 800 k Ω
lcd_swloadlk		LCD驱动倍压器使用1 k Ω 电阻替代200 Ω 与放电电容相连	

lcd_r_fastld		选择为LCD段码快速充电的电阻	0 = 30 kΩ 1 = 50 kΩ 2 = 200 kΩ 3 = 800 kΩ
sel_compint		选择内部比较器	1 = 选择使用
con_comp		控制比较器关闭模式	00 = 关闭 01 = 在两次单独测量直接关闭 10 = 整个充电时间和两次单独测量之间关闭 11 = 一直开启

Configreg_12: RAM 地址 60 EEPROM 字节 36 - 38

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lcd_pos																							

参数	推荐值	描述	设置
lcd_pos [23:0]	-	LCD 段码位置	

Configreg_13: RAM 地址 61 没有在EEPROM被镜像!

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lcd_segment																							

参数	推荐值	描述	设置
lcd_segment [23:0]	-	显示段码数字2~0	

Configreg_14: RAM 地址 62 没有在EEPROM被镜像!

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lcd_segment																							

参数	推荐值	描述	设置
lcd_segment [47:24]	-	显示段码数字5~3	

Configreg_15: RAM 地址 63 没有在EEPROM被镜像!

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lcd_segment																							

参数	推荐值	描述	设置
lcd_segment [55:48]	-	显示段码特殊字符 (1/3 duty和1/4 duty)	

Configreg_16: RAM 地址 64EEPROM 字节 39 - 41

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
														selqha							multio_sel			
1	0				1		0	0	1	0	0	0	1	0	1	0	1	0	0					

ien_noise_c10_lcd	lcd_spi_ena	lcd_charge	lcd_direct_drive	use_10kHz_lcd	em_fullbr_p	ena_nonlin	clkdivopr	ps_speed	fha_en	sense_discharge	lmultihit_en	sel_speed_rmh	run_la_cont
-------------------	-------------	------------	------------------	---------------	-------------	------------	-----------	----------	--------	-----------------	--------------	---------------	-------------

参数	推荐值	描述	设置
en_noise_c10_lcd	1	在从4MHz时钟分频出10kHz用于LCD电压充电泵的情况下，此位会加噪声给分频因数	1 = 激活
lcd_spi_ena	0	使用通过SPI接口传输的外部LCD驱动电路	1 = 通过SPI传输的外部LCD驱动使能 0 = 使用内部LCD驱动
lcd_charge[1:0]	-	选择LCD再次充电前周期数	0 = 每个周期都再充电一次 1 = 每1/2个周期再充电一次 2 = 每1/3个周期再充电一次 3 = 每1/4个周期再充电一次
lcd_direct_drive	-	用供电电压直接驱动LCD	1 = 激活
use_10kHz_lcd	1	LCD 电荷泵的时钟选择	0 = 电荷泵时钟从4 MHz时钟引出，分频值为480 ~512。此选择避免电荷泵与测量相混合，因此可以减少对测量的干扰 1 = 使用内部10kHz晶振

en_fullbr_p	-	Full bridge_P 模式使能。半桥端口管脚并联。Rds(on)减半, 漂移性能得到改善	1 = 使能
sense_discharge	1	设置比较器低通滤波电容的快速放电	1 = ena 使能 bled
multio_sel[3:0]	-	使用multio03 管脚来进行诊断	0 = multio 1 ... 11 = 内部使用 12 = 中断

Configreg_17: RAM 地址 65EEPROM 字节 42 - 44

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
spi_delay				lcd_pulsed				c10_div_lcd					lcd_spi_out			io_a		en_io5		en_io4		clks-syc	
0	1	1											0	0	0			1	0	1	0	1	0

n.c.

参数	推荐值	描述	设置
spi_delay[2:0]	3	设置LCD SPI主机模式接口时序	0 = 快(大约500ns) . . 7 = 慢(大约10 μs)
lcd_pulsed[3:0]	-	自动脉冲显示, 尤其是在太阳能电子秤应用中的待机情况下	00xx = 300 ms 闪烁期 01xx = 600 ms 闪烁期 10xx = 1 s 闪烁期 11xx = 2 s 闪烁期 xx00 = off xx01 = 100 ms 保持时间 xx10 = 200 ms 保持时间 xx11 = 400 ms 保持时间
c10_div_lcd[5:0]	-	从4MHz时钟分出10kHz时钟的分频因数 分频因数 = 480+c10_div_lcd	
lcd_spi_out[2:0]	1	当SPI输出时, 设置LCD管脚, 6个管脚, 6个管脚并联	xx1 = 开启LCD_COM1...LCD_COM4, LCD_SEG1, LCD_SEG2 作为 SCK x1x = 开启LCD_SEG3...LCD_SEG8 作为SSN 1xx = 开启LCD_SEG9...LCD_SEG14作为 SDO
io_a[1:0]	-	设置/读MULT_IO4 ~MULT_IO5 Output: 输出值, 可读回 Input: 读输入值	

en_io4[1:0]	2	MULT_I04端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
enio5[1:0]	2	MULT_I05端口定义	00 = 输出 01 = 上拉电阻输入 10 = 下拉电阻输入 11 = 输入
clksyc[1:0]	1	时钟同步的速度	

Configreg_18: RAM 地址 66EEPROM 字节 45 - 47

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult_NLK																							

目录

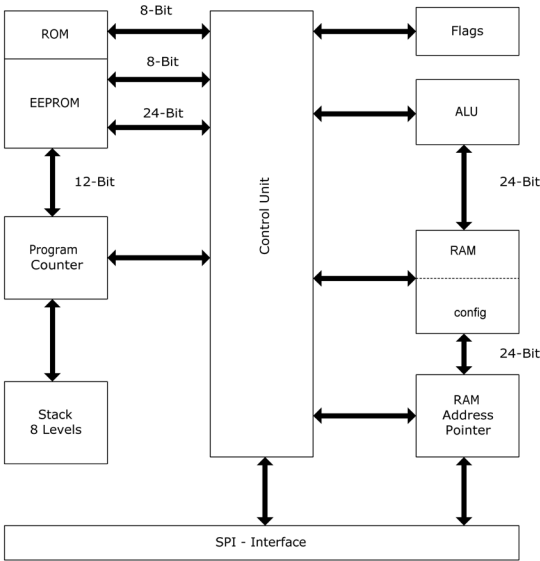
页面

6	中央处理单元 (CPU)	6-2
6.1	结构图	6-2
6.2	存储器结构	6-2
6.2.1	ROM 和 EEPROM	6-2
6.2.2	RAM 分配	6-3
6.2.3	RAM 的地址指针	6-4
6.2.4	算术逻辑单元 (ALU)	6-4
6.2.5	累加器	6-4
6.2.6	标志位	6-4
6.3	状态寄存器和结果寄存器	6-5
6.3.1	结果寄存器	6-5
6.3.2	状态寄存器	6-7
6.4	指令集	6-8
6.4.1	分支指令	6-8
6.4.2	算术操作	6-8
6.5	系统Reset, 休眠模式和自动配置	6-34
6.5.1	上电复位	6-34
6.5.2	看门狗重置	6-34
6.5.3	通过Pin 27进行外部复位	6-34
6.5.4	睡眠模式	6-35
6.5.5	CPU 时钟	6-36
6.5.6	看门狗计数器和单次转换计数器	6-36
6.5.7	计时器	6-36

6 中央处理单元 (CPU)

6.1 结构图

图 6.1 方框原理图



6.2 存储器结构
6.2.1 ROM 和 EEPROM

表 6.1

5119 ... 2048	ROM 程序存储单元
2047 ... 2000	用户程序存储单元 EEPROM 48 字节
1999 ... 48	程序存储单元 EEPROM 1.952 字节 程序进入
47...45 44...42 41...39	Config Reg 18 (镜像) Config Reg 17 (镜像) Config Reg 16 (镜像)
38...35	Config Reg 12 (镜像)
5...3 2...0	Config Reg 1 (镜像) Config Reg 0 (镜像)

ROM区域从2048地址开始。 PICOSTRAIN测量内部所需要的所有计算程式都存储在这里。还有其他一些在称重应用中经常会用到的函数也存储在这里，例如十进制到七段码的转换程序，求平均值程序，线性修正程序，hysterisis程序等等。这些函数可以通过EEPROM中的程序访问。同样在访问ROM中的函数后可以通过配置程序跳回到EEPROM的用户代码中。

EEPROM总共有2048字节的空间，被分成4块，每块521字节。程序存储从48地址开始，占据1952字节。每次从ROM跳到EEPROM时，都是从48地址开始。EEPROM中的程序通过分析Register22中存储的状态信息可以找出跳转的原因。

上电复位时，PS081会进行自动配置，EEPROM中的低48字节被预留起来存储自动配置数据。3个连续的字节合在一起组成一个24位的字。因此有16个24位的字能够通过程序代码读取。它们被用于配置寄存器的0~12和寄存器16~18。在上电复位中它们被复制到RAM的48地址到66地址。

EEPROM的2000到2047单元是可以自由读写的。通过putepr和getepr（地址0~15）操作，处理器可以把数据写入这些单元或者从这些单元中读取数据。这些单元可用来存储校准数据，以防在校准过程中数据丢失。

6.2.2 RAM 分配

表 6.2

127 ... 112	User RAM 127 ... User RAM 112	有可能会被一些ROM函数所占用，例如median滤波
111 ... 100 99...98 97 ... 67	User RAM（应用于转换器和ALU） LCD段的内容（外部） User RAM（应用于转换器和ALU）	
66 ... 48	Config Reg 18 ... Config Reg 0	
47 ... 32	User RAM 47 ... User RAM 32	
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	- - Timer I/O 状态 被按下 I/O 状态 下降沿 I/O 状态 上升沿 UBATT CAL HB1+ 标志位 (p1-p2)/p2 $HB0 = (A-B) + (C-D) + (\dots) / (A+B) + (C+D) + (\dots) *$ $HB4 = (G-H) / (G+H)$ $HB3 = (E-F) / (E+F)$ $HB2 = (C-D) / (C+D)$ $HB1 = (A-B) / (A+B)$	
15 ... 0	User RAM 15 ... User RAM 0	

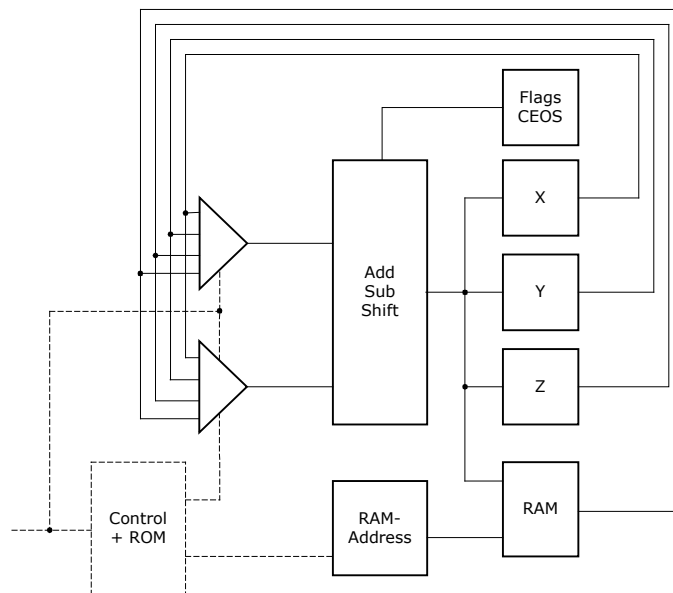
参数A...F代表不同端口的放电时间，详细介绍请参考6.2.4章节

6.2.3 RAM 的地址指针

RAM有它自己的地址总线，包含127个的地址。每个RAM单元有24位，其宽度与ALU寄存器的宽度相同。一个单独的RAM地址通过RAM地址指针编址后映射到ALU，可用作第四个累加寄存器。改变RAM的地址指针不会影响存储在其所指地址中的数据。可通过ramadr, incramadr等不同的操作指令来改变RAM的地址指针所指的地址。

6.2.4 算数逻辑单元 (ALU)

图 6.2 ALU结构图



6.2.5 累加器

ALU有3个24位的累加器，X、Y和Z。RAM可以由RAM的地址指针来编地址，被编址的RAM的单元可用作第四个累加器。单个RAM地址可由RAM地址指针映射到ALU，所以加在一起总共有4个累加器。所有的转移操作（move, swap）和算术操作（shift, add, mult24...）对每一个累加器都适用。

6.2.6 标志位

处理器对于每个操作都控制4个标志位。在对每一个累加器（包括 RAM）读操作时 不等于（not-equal）和符号标志会被置位。另外，进位和溢出标志位在计算的情况下被置位（Add/Sub/shiftR）。可以通过jump命令来跨越每个标志。

6.2.6.1 进位

显示了一个加法或减法中的进位。与移位操作一起时（shiftL, rotR 等）它显示的为延迟位。

6.2.6.2 非零

这个标志当有新的结果为 0 写到累加器 (add, sub, move, swap 等) 的时候被置位。

6.2.6.3 符号

这个符号标志当有新的结果被写到累加器 (add, sub, move, swap 等) 以及最高位 (MSB) 为 1 时被置位。

6.2.6.4 溢出

指示两个以补码形式表示的数在进行加法或是减法过程中是否有溢出。

6.3 状态寄存器和结果寄存器

6.3.1 结果寄存器

存储测量结束时RAM结果寄存器中的内容:

ram=16	: $HB1 = (A-B) / (A+B)$	HB1 未补偿
ram=17	: $HB2 = (C-D) / (C+D)$	HB2 未补偿
ram=18	: $HB3 = (E-F) / (E+F)$	HB3 未补偿
ram=19	: $HB4 = (G-H) / (G+H)$	HB4 未补偿
ram=20	: $HB0 = (A-B) + (C-D) + (..) /$ $(A+B) + (C+D) + (..)$	HB0 补偿后的和
ram=21	: $TMP = (p1-p2) / p2$	温度
ram=22	: Status	See 6.3.2
ram=23	: HB1+	时间测量TDC (SG_A1, Pin11)
ram=24	: CAL	TDC分辨率
ram=25	: UBATT	测量出的供电电压
ram=26	: Status_F	表示单个I/O端口或者两个以上I/O端口边沿由高到低
ram=27	: Status_R	表示单个I/O端口或者两个以上I/O端口边沿由低到高
ram=28	: Status_P	表示单个I/O端口或者两个以上I/O端口为高
ram=29	: Timer	
ram=30, 31	: NC	空位, 可暂时被使用但是测量中会被覆盖
x-Accu	: HB0	RAM值 = 20
y-Accu	: Temp	RAM值 = 21
z-Accu	: Flags	RAM值 = 63
ramadr	: 0	RAM 地址指针值

描述:

- A : 在 SG_A1 端口进行放电时间测量
 B : 在 SG_A2 端口进行放电时间测量

C :	在 SG_B1 端口进行放电时间测量
D :	在 SG_B2 端口进行放电时间测量
E :	在 SG_C1 端口进行放电时间测量
F :	在 SG_C2 端口进行放电时间测量
G :	在 SG_D1 端口进行放电时间测量
H :	在 SG_D2 端口进行放电时间测量
P1:	在 TMP_1 端口进行放电时间测量
P2:	在 TMP_2 端口进行放电时间测量

格式:

HB1:	结果是100ppm的倍数, $HB1/100 = \text{结果}$, 单位是ppm
HB2:	结果是100ppm的倍数, $HB2/100 = \text{结果}$, 单位是ppm
HB3:	结果是100ppm的倍数, $HB3/100 = \text{结果}$, 单位是ppm
HB4:	结果是100ppm的倍数, $HB4/100 = \text{结果}$, 单位是ppm
HB0:	结果是100ppm的倍数, $HB0/100 = \text{结果}$, 单位是ppm
TMP:	结果(Tmp) $= TMP/2^{24}$
Status:	参见上述说明
HB1+:	结果 $(HB1+)/ns = 250 * HB1+ / 2^{14}$ [4MHz 时钟]
CAL:	结果 $(Cal)/ps = 250,000 / CAL$ [4MHz 时钟]
UBATT:	结果 $(UBATT)/V = 2.0 + 1.6 * UBATT / 64$

HB1, HB2, HB3, HB4, HB0, TMP是以2的补码形式给出的, MSB最高位=1表示此数是负值。
乘以 -2^{24} 就可以得到正值。

注释:

以一个具有标准性能的传感器 (2mV/V) 为例, 其阻值的变化是0.2%, 也就是说阻值为1000Ω的传感器其变化为2Ω。0.2%的变化相当于2000ppm。受内部算法及准确性等原因影响, 那么结果是以 $100 \times 2000\text{ppm}$ (=200000ppm) 的形式给出的。请注意, 此寄存器中的值不仅和传感器的灵敏度有关, 而且和PS081中设置的Mult_HBx有关。以上例子中 Mult_HBx=1。

例子:

传感器为1.5mV/V, Picostrain连线, Mult_HBx=1:

1.5mV/V=1500ppm → 在最大应变时PS081中的结果为: 150,000 (0×0249F0)

传感器为2mV/V，惠斯通电桥连线，Mult_HBx=1:

2mV/V在惠斯通电桥中相当于1.333mV/V=1333ppm（由于应变惠斯通桥灵敏度的损失）——>
在最大应变时PS081中的结果为：133333（0×0208D5）

传感器为1mV/V，Picostrain连线，Mult_HBx=4:

1mV/V=1000ppm ——> 在最大应变时PS081中的结果为：400000（0×061A80）

6.3.2 状态寄存器

表 6.3 状态寄存器

位	描述
Status[23]= flg_io3_mio	管脚26(32)
Status[22]= flg_io2_sck	管脚20(25)
Status[21]= flg_io1_sdi	管脚19(24)
Status[20]= flg_io0_sdo	管脚18(23)
Status[19]= flg_rstpwr	1 = 跳转到EEPROM，由上电复位引起
Status[18]= flg_rstssn	1 = 跳转到EEPROM，由按下按钮引起
Status[17]= flg_wdtalt	1 = 跳转到EEPROM，由看门狗中断引起
Status[16]= flg_endavg	1 = 测量完成后跳转到EEPROM
Status[15]= flg_intav0	1 = 在睡眠模式下跳转到EEPROM
Status[14]= flg_ub_low	1 = 低电压
Status[13]= flg_errtdc	1 = TDC出错
Status[12]= flg_pslock[1]	1 = 移相器锁
Status[11]= flg_pslock[0]	1 = 移相器锁
Status[10]= flg_errprt	1 = 应变测量端口出错
Status[09]= flg_timeout	1 = TDC溢出
Status[08]= error_dspclk	1 = TDC 和 DSP 冲突
Status[07]= flg_io3_mio_r	1 = Pin26(32) 接收到上升沿, 0 = 无上升沿
Status[06]= flg_io2_sck_r	1 = Pin20(25) 接收到上升沿, 0 = 无上升沿
Status[05]= flg_io1_sdi_r	1 = Pin19(24) 接收到上升沿, 0 = 无上升沿
Status[04]= flg_io0_sdo_r	1 = Pin18(23) 接收到上升沿, 0 = 无上升沿
Status[03]= flg_io3_mio_f	1 = Pin26(32) 接收到下降沿, 0 = 无下降沿
Status[02]= flg_io2_sck_f	1 = Pin20(25) 接收到下降沿, 0 = 无下降沿
Status[01]= flg_io1_sdi_f	1 = Pin19(24) 接收到下降沿, 0 = 无下降沿
Status[00]= flg_io0_sdo_f	1 = Pin18(23) 接收到下降沿, 0 = 无下降沿

在括号中的Pin管脚号 = 裸片的管脚号

在状态寄存器（RAM26~28）中可以查询出输入的状态。详细介绍请参照4.4.4章节（页4-17）。

6.4 指令集

PS081完整指令集包括94个核心指令，这些核心指令对应唯一的被CPU编码过的op-code操作码。此外还有仿真指令，例如 no2lcd，由汇编程序自动代替，调用ROM代码中的子程序。指令集的多样性和综合性允许写入非常全面而且大量的程序而仅用 2K EEPROM。

6.4.1 分支指令

在代码中，有3个跳转指令：

- Jump指令：后接12位绝对地址，这个地址需在整個地址空间內。
- Branch指令：后接8位相对地址，这个相对地址的范围为-128到+127。
- Skip指令：向前跳3个操作码（3~15字节）

汇编程序把JUMP 和BRANCH一起放到了GOTO指令中。

只使用绝对跳转而不附带任何条件也可以跳转到子程序中。

6.4.2 算数操作

RAM 是 24 位字的结构。所有的指令都是 2 进制为基础。一次算术操作命令结合了两个累加器然后将结果写入第一次提到过的累加器中。RAM 地址指针显示了 RAM 地址，这个地址的处理方式与累加器相同。每一次在累加器的操作都会影响到 4 个标志位。标志位以最后一次操作为准。

表 6.4 指令集

简单算术运算指令	复杂运算指令	移位和旋转指令	RAM操作指令
abs add compare compl decr getflag incr sign sub	div24 divmod mult24 mult48	clrC rotl rotR setC shiftL shiftR	clear decramadr incramadr move ramadr swap

逻辑运算指令	位操作指令	LCD 显示指令	EEPROM 读取指令
and eor nor invert nand nor or	bitclr bitinv bitset	clrLCD dez2lcd newlcd no2lcd no2LcdAccu setLCD spi2lcd	equal getepr putepr useEprInit

无条件跳转指令	标志跳转指令	其它指令
goto	skip	clk10kHz
gotoBitC	skipBitC	clrwdt
gotoBitS	skipBitS	gr2st
gotoCarC	skipCarC	hysterises
gotoCarS	skipCarS	initAvg
gotoEQ	skipEQ	initTDC
gotoNE	skipNE	median
gotoNeg	skipNeg	newcyc
gotoOvrC	skipOvrC	nop
gotoOvrS	skipOvrS	rollAvg
gotoPos	skipPos	round
jsub		ssnPulse
jsubret		ssnSet
		stop

abs 寄存器的绝对值

句法:	abs p1
参数:	p1 = ACCU [x, y, z, r]
算法:	$p1 := p1 $
影响标志位:	C O S Z
字节:	2
循环:	2
描述:	寄存器值的绝对值
类别:	简单算术运算

add 加法

句法:	add p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或者 24-位数
算法:	$p1 := p1 + p2$
影响标志位:	C O S Z
字节:	2 (p2 = ACCU) 4 (p2 = number)
循环:	2 (p2 = ACCU) 4 (p2 = number)
描述:	两个寄存器相加或者一个常数和寄存器相加
类别:	简单算术运算

and	逻辑与
句法:	and p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或者 24-位
算法:	p1 := p1 AND p2
影响标志位:	S Z
字节:	2 (p2 = ACCU) 5 (p2 = number)
循环:	3 (p2 = ACCU) 6 (p2 = number)
描述:	两个寄存器的逻辑AND 或者寄存器和常数的逻辑 AND
类别:	逻辑运算

bitclr	清除位
句法:	bitclr p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = number 0 到 23
算法:	p1:=p1 and not (1<<p2)
影响标志位:	S Z
字节:	2
循环:	2
描述:	在目标寄存器中清除其中一位
类别:	按位运算 (Bitwise)

bitinv	反相位
句法:	bitinv p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = number 0 到 23
算法:	p1:=p1 eor (1<<p2)
影响标志位:	S Z
字节:	2
循环:	2
描述:	在目标寄存器中翻转其中一位
类别:	按位运算 (Bitwise)

bitset	设置位
句法:	bitset p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = number 0 到 23
算法:	p1:=p1 or (1<<p2)
影响标志位:	S Z
字节:	2
循环:	2
描述:	在目标寄存器中设置其中一位
类别:	按位运算 (Bitwise)

clear	清除寄存器
句法:	clear p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := 0
影响标志位:	S Z
字节:	1
循环:	1
描述:	将所被寻址的寄存器清 0
类别:	RAM 控制

clk10khz	时钟源10 kHz
句法:	clk10khz p1
参数:	p1 = number 0 or 1
算法:	–
影响标志位:	–
字节:	2
循环:	3
描述:	将处理器的时钟源改变到10kHz。处理器的时钟源被从4MHz切换到低速10kHz时钟。10 kHz时钟对于温度和供电电压变化很稳定。如果p1被设置为1, 那么10kHz时钟则开启, 如果 p1 == 0 那么10 kHz 时钟将被关闭。
类别:	其它

clrC	清除标志位
句法:	clrC
参数:	–
算法:	–
影响标志位:	C 0
字节:	1
循环:	1
描述:	清除进位和溢出标志
类别:	移动和旋转

clrLCD	清除LCD
句法:	clrLCD
参数:	–
算法:	–
影响标志位:	–
字节:	1
循环:	子程序调用
描述:	把LCD寄存器61~62清零. 此指令与newlcd指令结合使用, 可关闭所有LCD段码。在ROM代码子程序中被调用。 在汇编程序中, 此指令会被编译成相应的跳转指令。
类别:	LCD 显示

clrwdt	清除看门狗
句法:	clrwdt
参数:	–
算法:	–
影响标志位:	–
字节:	2
循环:	
描述:	清除看门狗。这个操作码用于在程序运行结束后清出看门狗定时器。在‘stop’之前应用此操作码。
类别:	其它
compare	比较两个数值
句法:	compare p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或者 24-位数
算法:	---:=p2-p1 仅仅标志位被改变而不是寄存器
影响标志位:	C O S Z
字节:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
描述:	通过相减比较2个寄存器 通过相减比较寄存器和一个常数 标志位根据相减的结果而改变，但是寄存器的内容本身不变
类别:	简单算术运算
compl	2进制补码
句法:	compl p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := - p1 = not p1 + 1
影响标志位:	S Z
字节:	2
循环:	2
描述:	寄存器的2进制补码
类别:	简单算术运算
decr	自减
句法:	decr p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := p1 - 1
影响标志位:	C O S Z
字节:	1
循环:	1
描述:	寄存器自减
类别:	简单算术运算

decramadr	RAM 地址指针自减
句法:	decramadr
参数:	–
算法:	–
影响标志位:	–
字节:	1
循环:	1
描述:	将ram地址指针减少1
类别:	RAM 操作

dez2lcd	10进制到段码转换																																												
句法:	dez2lcd p1																																												
参数:	p1 = ACCU [x, y, z, r]																																												
算法:	—																																												
影响标志位:	—																																												
字节:	2																																												
循环:	3																																												
描述:	转换在寄存器p1中的10进制代码到7段码。从寄存器p1的0-9的十进制数被转换到标准的7段码(数字a-h)。这个操作码可以用于高级的LCD 转换程序当中，当no2lcd不够的时候 <table><tr><td>dec</td><td></td><td>hgfe</td><td>dcba</td></tr><tr><td>0</td><td>→</td><td>0b00111111</td><td>=0x3F</td></tr><tr><td>1</td><td>→</td><td>0b00000110</td><td>=0x06</td></tr><tr><td>2</td><td>→</td><td>0b00111011</td><td>=0x3B</td></tr><tr><td>3</td><td>→</td><td>0b01001111</td><td>=0x4F</td></tr><tr><td>4</td><td>→</td><td>0b01100110</td><td>=0x66</td></tr><tr><td>5</td><td>→</td><td>0b01101101</td><td>=0x6D</td></tr><tr><td>6</td><td>→</td><td>0b01111101</td><td>=0x7D</td></tr><tr><td>7</td><td>→</td><td>0b00000111</td><td>=0x07</td></tr><tr><td>8</td><td>→</td><td>0b01111111</td><td>=0x7F</td></tr><tr><td>9</td><td>→</td><td>0b01101111</td><td>=0x6F</td></tr></table>	dec		hgfe	dcba	0	→	0b00111111	=0x3F	1	→	0b00000110	=0x06	2	→	0b00111011	=0x3B	3	→	0b01001111	=0x4F	4	→	0b01100110	=0x66	5	→	0b01101101	=0x6D	6	→	0b01111101	=0x7D	7	→	0b00000111	=0x07	8	→	0b01111111	=0x7F	9	→	0b01101111	=0x6F
dec		hgfe	dcba																																										
0	→	0b00111111	=0x3F																																										
1	→	0b00000110	=0x06																																										
2	→	0b00111011	=0x3B																																										
3	→	0b01001111	=0x4F																																										
4	→	0b01100110	=0x66																																										
5	→	0b01101101	=0x6D																																										
6	→	0b01111101	=0x7D																																										
7	→	0b00000111	=0x07																																										
8	→	0b01111111	=0x7F																																										
9	→	0b01101111	=0x6F																																										
类别:	LCD 显示																																												

div24	带符号除法 24 位
句法:	div24 p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r]
算法:	$p1 := (p1 \ll 24) / p2$ (if $ p1 < p2/2 $)
影响标志位:	S & Z of p1
字节:	2
循环:	20
描述:	两个寄存器的带符号除法 24 位的两个寄存器除法，结果被分配到p1
类别:	复杂算术运算

divmod	带符号模除法
句法:	divmod p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r]
算法:	p1 := p1 / p2 and p2 := p1 % p2
影响标志位:	S Z
字节:	2
循环:	
描述:	两个寄存器的带符号模除法运算 两个寄存器24高位除法，结果被分配到p1。余数被分配到p2。
类别:	复杂算术运算

eor	异或
句法:	eor p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或24-位数
算法:	p1 := p1 xor p2 按位异或 0 / 0 和 1 / 1 返回 0 按位异或 0 / 1 和 1 / 0 返回 1
影响标志位:	S Z
字节:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
循环:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
描述:	两个给定寄存器的逻辑XOR (eXclusive OR, 不相等) 常数和寄存器的逻辑XOR (eXclusive OR, 不相等)
类别:	逻辑运算

eorn	异或非
句法:	eorn p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或24-位数
算法:	p1 := p1 xnor p2 按位运算 0 / 0 和 1 / 1 返回 1 按位运算 0 / 1 和 1 / 0 返回 0
影响标志位:	S Z
字节:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
循环:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
描述:	两个给定寄存器的逻辑XNOR (eXclusive NOR, 相等) 常数和寄存器的逻辑 XNOR (eXclusive NOR, 相等)
类别:	逻辑运算

equal	写3 Bytes 到 EEPROM
句法:	equal p1
参数:	p1 = 24-Bit number
算法:	-
影响标志位:	-
字节:	3
循环:	
描述:	写3bytes 到 (p1) EEPROM 的配置寄存器。Equal 操作码被用来直接写3 个bytes 到EEPROM 寄存器的配置数据当中。因此这个操作码在汇编程序开始时将被直接应用16次通过p1 给配置数据, 比如像 ‘putep1’ 这个操作码。EEPROM 的配置是在从0..47byte空间中完成, 与16 x 24 位的寄存器结合。从48byte 开始, 用户可以写自己的代码到EEPROM。应用这个操作码, 可以写如您自己的配置寄存器代替标准的配置。
类别:	EEPROM 操作

getep1	获得 EEPROM 内容
句法:	getep1 p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := EEPROM 寄存器内容(通过RAM 地址指针寻址)
影响标志位:	S Z
字节:	1
循环:	6
描述:	将 EEPROM 内容给到寄存器当中。被指配的结果寄存器 p1 通过RAM 地址指针的操作获得EEPROM 寄存器的内容。这个操作码需要在程序计数堆栈中一个临时的空间(详细说明见下面)。
类别:	EEPROM 操作

getflag	置位S 和 Z 标志
句法:	getflag p1
参数:	p1 = ACCU [x, y, z, r]
算法:	signum := set if p1 < 0 notequalzero := set if p1 <> 0
影响标志位:	S Z
字节:	1
循环:	1
描述:	置位符号和非零标志位根据被寻址的寄存器, 寄存器的内容不受影响。
类别:	简单算数运算

goto	无条件跳转
句法:	goto p1
参数:	p1 = JUMPLABEL
算法:	PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	无条件跳转。程序计数器被设置到目标地址上。目标地址通过应用一个 jumplabel 给出。参见例程部分来了解如何引入一个 jumplabel。
类别:	无条件跳转

gotoBitC	位清零跳转
句法:	gotoBitC p1, p2, p3
参数:	p1 = ACCU [x, y, z, r] p2 = NUMBER [0..23] p3 = JUMPLABEL
算法:	if (寄存器p1 的位p2 == 0) PC := p3
影响标志位:	–
字节:	3
循环:	4
描述:	位清零跳转。程序计数器将会被设置到目标地址如果在寄存器p1 的所选择位被清零的时候。目标地址通过应用一个 jumplabel 给出。参见例程部分来了解如何引入一个 jumplabel。
类别:	位运算

gotoBitS	当所选位被置时跳转
句法:	gotoBitS p1, p2, p3
参数:	p1 = ACCU [x, y, z, r] p2 = NUMBER [0..23] p3 = JUMPLABEL
算法:	if (寄存器p1 的位p2 == 1) PC := p3
影响标志位:	–
字节:	3
循环:	4
描述:	当所选位被置时跳转。程序计数器将会被设置到目标地址如果在寄存器p1 的所选择位被置位的时候。目标地址通过应用一个 jumplabel 给出。参见例程部分来了解如何引入一个 jumplabel。
类别:	位运算

gotoCarC	进位清零跳转
句法:	gotoCarC p1
参数:	p1 = JUMPLABEL
算法:	if (carry == 0) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当进位清零时跳转。程序计数器将会被设置到目标地址当进位标志清零的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel。
类别:	位跳转

gotoCarS	进位被置位时跳转
句法:	gotoCarS p1
参数:	p1 = JUMPLABEL
算法:	if (carry == 1) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	进位被置位时跳转。程序计数器将会被设置到目标地址当进位标志被置位的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel。
类别:	位跳转

gotoEQ	等于零跳转
句法:	gotoEQ p1
参数:	p1 = JUMPLABEL
算法:	if (Z == 0) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当等于零的时候跳转。程序计数器将会被设置到目标地址当前面的结果为零的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel。
类别:	位跳转

gotoNE	非零时跳转
句法:	gotoNE p1
参数:	p1 = JUMPLABEL
算法:	if (Z == 1) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当结果不等于零时跳转。程序计数器将会被设置到目标地址当前面的结果为不等于零的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel。
类别:	位跳转

gotoNeg	负数值跳转
句法:	gotoNeg p1
参数:	p1 = JUMPLABEL
算法:	if (S == 1) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	结果为负值时跳转。程序计数器将会被设置到目标地址当前面的结果为负值的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel
类别:	位跳转

gotoOvrC	溢出标志清零跳转
句法:	gotoOvrC p1
参数:	p1 = JUMPLABEL
算法:	if (0 == 0) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当溢出标志位清零的时候跳转。程序计数器将会被设置到目标地址当溢出标志清零的时候。目标地址通过应用一个jumplabel 给出。参见例程部分来了解如何引入一个jumplabel。
类别:	位跳转

gotoOvrS	溢出标志位置位跳转
句法:	gotoOvrS p1
参数:	p1 = JUMPLABEL
算法:	if (0 == 1) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当溢出标志位置位的时候跳转。程序计数器将会被设置到目标地址当溢出标志被置位的时候。目标地址通过应用一个jump1abel 给出。参见例程部分来了解如何引入一个jump1abel。
类别:	位跳转

gotoPos	正值跳转
句法:	gotoPos p1
参数:	p1 = JUMPLABEL
算法:	if (S == 0) PC := p1
影响标志位:	–
字节:	2 (相对跳转) 3 (绝对跳转)
循环:	3 (相对跳转) 4 (绝对跳转)
描述:	当结果为正值的时候跳转。程序计数器将会被设置到目标地址当结果为正数的时候。目标地址通过应用一个jump1abel 给出。参见例程部分来了解如何引入一个jump1abel。
类别:	位跳转

gr2st	Gram 到 Stone 的单位转换
句法:	gr2st p1
参数:	p1 = ACCU [x]
算法:	p1 = stone (p1)
影响标志位:	–
字节:	3
循环:	185
描述:	将100mg的倍数转换成以stone为单位的数值。结果的2位以stone给出，剩下的位以磅给出。那么这个值可以直接通过no21cd显示到LCD。
类别:	其他

hysteresis	Hysteresis
句法:	hysteresis p1, p2, p3
参数:	p1 = ACCU [x] p2 = NUMBER [half scale division] p2 = NUMBER [half scale division + half hysteresis]
算法:	p1 = hysteresis (p1)
影响标志位:	-
字节:	10
循环:	子程序调用
描述:	这个子程序提供了一个 hysteresis 的功能. 通过参数 p3 你可以定义一个数值得范围, 那么这个范围是对于当前侧俩功能值得一个”容忍”范围。换句话说, 尽管这个时候分度应该加一了, 但是由于这个功能还将显示当前的分度, 而不做加一处理。这个操作码包括了一个近似值的操作。这个近似值的功能是通过P2设置的。请同时参见这页下面的解释。
类别:	其他
incr	自增
句法:	incr p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := p1 + 1
影响标志位:	C O S Z
字节:	1
循环:	1
描述:	寄存器增加1
类别:	简单算术运算
incramadr	RAM 地址自增
句法:	incramadr
参数:	-
算法:	-
影响标志位:	-
字节:	1
循环:	1
描述:	RAM地址指针加1
类别:	RAM 操作
initAvg	滑动平均值的初始化
句法:	initAvg p1, p2
参数:	p1 = ACCU[x] p2 = 数值从3到17
算法:	-
影响标志位:	-
字节:	2
循环:	子函数调用
描述:	初始化滑动平均值。 p1设置了默认计算的方式。p2 则设置了多少个结果做滑动平均。 在实际操作中这个函数是在 ROM代码中被调用的。Assembler将这个命令转换为一个相应得跳转命令。
类别:	其他

initTDC	初始化TDC
句法:	initTDC
参数:	–
算法:	–
影响标志位:	–
字节:	2
循环:	3
描述:	TDC (time-to-digital converter)的初始化操作。应该在配置寄存器结束后发送。这个初始化操作将不会改变寄存器的配置。
类别:	其它

invert	按位翻转
句法:	invert p1
参数:	p1 = ACCU [x, y, z, r]
算法:	p1 := not p1
影响标志位:	S Z
字节:	2
循环:	2
描述:	对寄存器按位翻转
类别:	逻辑运算

jsub	无条件跳转
句法:	jsub p1
参数:	p1 = JUMPLABEL
算法:	PC := p1
影响标志位:	C 0 S Z
字节:	3
循环:	4
描述:	无条件跳转到子程序中。程序计数器通过jumplabel 给出的地址所增加。子程序一直运行直到 关键字 ‘jsubret’ 出现。然后将会跳回来执行jsub-call 下面的命令。这个操作码在程序计数堆栈中需要一个临时的空间（解释见下面）。
类别:	无条件跳转

jsubret	从子程序中返回
句法:	jsubret
参数:	–
算法:	PC := PC from jsub-call
影响标志位:	–
字节:	1
循环:	3
描述:	R从子程序中返回。子程序可以通过呼叫 ‘jsub’ 调用然后通过呼叫jsubret 来退出。程序将继续执行jsub-call 后面的命令。必须通过 jsubret 关闭子程序 - 否则将不会跳回。
类别:	无条件跳转

median	Median 滤波（求平均中值）
句法:	median p1, p2
参数:	p1 = ACCU [x] p2 = NUMBER [3 to 15, uneven]
算法:	p1 = median [p1+r(32)+r(33)+...+r(32+p2-2)]
影响标志位:	S Z (of result p1)
字节:	7
循环:	子函数调用
描述:	应用median滤波，滤波深度可以自己设置。所被加的数是给到Accu X当中。那么这个数定义了滤波的深度，仅能够设置基数。为了应用这个操作码，需要对相应的RAM-单元进行初始化，这个初始化可以通过initAvg来完成
类别:	其他
move	移动
句法:	move p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或者 24-常数
算法:	p1 := p2
影响标志位:	S Z
字节:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
描述:	将p2 的内容移动到p1 (p1=ACCU, p2=ACCU) 将常数移动到p1 (p1=ACCU, p2=NUMBER)
类别:	RAM 操作
mult24	带符号的24 位乘法
句法:	mult24 p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r]
算法:	p1 := (p1 * p2) >> 24
影响标志位:	S & Z of p1
字节:	2
循环:	30
描述:	带符号的两个寄存器的乘法运算比如mult48。但是只有两个寄存器的高24位相乘，结果存在p1当中。
类别:	复杂算术运算
mult48	带符号的48 位乘法
句法:	mult48 p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r]
算法:	p1, p2 := p1 * p2
影响标志位:	S & Z of p1
字节:	2
循环:	30
描述:	带符号的两个寄存器的乘法运算 高24位结果存储到p1 低24位结果存储到p2
类别:	复杂算术运算

nand	逻辑与非
句法:	nand p1, p2
参数:	p1 = ACCU [x, y, z, r] p1 = ACCU [x, y, z, r] 或者24-位常数
算法:	p1 := p1 nand p2 仅在相应位为1 / 1 的时候返回 0
影响标志位:	S Z
字节:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
循环:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
描述:	两个给定寄存器的逻辑 NAND (反向 AND) 寄存器和常数的逻辑NAND (反向AND)
类别:	逻辑运算
newcyc	Start TDC
句法:	newcyc
参数:	–
算法:	–
影响标志位:	–
字节:	2
循环:	3
描述:	TDC测量开始。 这个操作码在PS08 配置寄存器和初始化结束后开始新的测量循环。一般这是通过PS08 的rom 程序本身完成，但是当用户指定的复位被应用得时候，这个操作码就有一定的用途了。
类别:	其它
newlcd	载入新的 LCD 数据
句法:	newlcd
参数:	–
算法:	–
影响标志位:	–
字节:	2
循环:	3
描述:	载入新的LCD 数据。新的段码数据从寄存器61-63 写入 LCD 驱动。 刷新显示器。
类别:	LCD 显示
no2lcd	转换6 个数字到 LCD 码
句法:	no2lcd p1, p2
参数:	p1 = ACCU[x] p2 = number 1 to 6
算法:	–
影响标志位:	–
字节:	3
循环:	呼叫子程序
描述:	将寄存器 x 中的6 位十进制数转换为7-段码。小数点的位置是由寄存器y中的p2数 (0..5) 来决定的。小数点前面的零将会被清除。 转换的结果直接写到 LCD 寄存器61-62 与 ‘newlcd’ 一同应用。 实际上这个操作码是在调用ROM 中的一个子程序。PS081汇编将这个命令转换成相应的跳转命令。
类别:	LCD 显示

no2lcdAccu	转换6 个数字到 LCD 码
句法:	no2lcd p1, p2
参数:	p1 = ACCU[x] p1 = ACCU[Y]
算法:	–
影响标志位:	–
字节:	3
循环:	呼叫子程序
描述:	把寄存器X中的十进制数（0~6）转换成7段代码。小数点的位置由p2中的数据决定。小数点前的0全部被清除。 此转换结果直接被写入LCD寄存器61~62。 此操作码与, newlcd ‘搭配使用。
类别:	LCD 显示

nop	无操作
句法:	–
参数:	–
算法:	–
影响标志位:	–
字节:	1
循环:	1
描述:	空间占用码或者时间调整(没有功能)
类别:	其它

nor	逻辑 NOR
句法:	nor p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或24-位常数
算法:	p1 := p1 nor p2 仅在相应位为 0 / 0 时返回1
影响标志位:	S Z
字节:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
循环:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
描述:	两个寄存器的逻辑NOR 运算 (反向 OR) 寄存器和常数的逻辑NOR 运算(反向OR)
类别:	逻辑运算

or	逻辑 OR
句法:	or p1, p2
参数:	p1 = ACCU [x, y, z, r] p2 = ACCU [x, y, z, r] 或 24-位常数
算法:	p1 := p1 or p2 仅当相对应位为 0 / 0 时返回0
影响标志位:	S Z
字节:	2 (p1=ACCU, p2=ACCU) 5 (p1=ACCU, p2=NUMBER)
循环:	3 (p1=ACCU, p2=ACCU) 6 (p1=ACCU, p2=NUMBER)
描述:	两个寄存器的逻辑OR运算 寄存器和常数的逻辑OR运算
类别:	逻辑运算

puteptr	将寄存器内容放入EEPROM
句法:	puteptr p1
参数:	p1 = ACCU [x, y, z, r]
算法:	EEPROM 寄存器 (通过ram地址指针寻址) := p1
影响标志位:	-
字节:	4
循环:	65536 = ~50ms
描述:	将寄存器内容放入EEPROM。 将所指定地址的寄存器p1 移动到EEPROM (EEPROM 寄存器地址通过 ram地址指针来设置)。 EEPROM字节2000-2047通过'puteptr' 进入, 设置RAM地址指针到地址0-15。这个操作码在程序计数堆栈中需要一个临时空间 (解释见下面)。 'puteptr' 不可以与skip-操作码同时使用, 因为这个操作码的执行时间太长 (大约: 50ms)
类别:	EEPROM 操作

ramadr	设置 RAM 地址指针
句法:	ramadr p1
参数:	p1 = 5-位常数
算法:	-
影响标志位:	-
字节:	1
循环:	1
描述:	设置指针到 ram 地址 (范围: 0..65)
类别:	RAM 操作

rollAvg	滑动平均值的计算
句法:	rollAvg p1,p2
参数:	p1 = ACCU[x] p2 = number from 3 to 17
算法:	ACCU[y] = ACCU[x] old p1 = rolling average result
影响标志位:	-
字节:	2
循环:	子函数调用
描述:	把p1中的值提供给求滑动平均值子程序。p2定义了求滑动平均值中数据的个数。从滑动平均值中掉出的数值被存在RAM地址13中。结果存在x ACCU中。先前的结果存储在y ACCU中。 实际上此命令是调用了ROM中的子程序。汇编程序会把此命令转换成相应的跳转命令。
类别:	其它

rotL	向左旋转
句法:	rotL p1(,p2)
参数:	p1 = ACCU [x, y, z, r] p2 = 4-位常数或无
算法:	p1 := p1<< 1+ carry; carry:=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) rotL p1 (in case rotL p1,p2)
影响标志位:	C O S Z (of the last step)
字节:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
描述:	向左旋转 p1 --> 向左移动 p1 寄存器, 将进位寄存器的位给LSB, MSB 被放在进位寄存器中。 将p1 寄存器带有进位向左旋转p2 次 -->将p1 寄存器带有进位向左旋转p2次, 在每次旋转中将进位放到LSB 中, 将MSB 位放到进位中。
类别:	移动 & 旋转

rotR	向右旋转
句法:	rotR p1(, p2)
参数:	p1 = ACCU [x, y, z, r] p2 = 4-Bit 位常数或无
算法:	p1 := p1>> 1+ carry; carry:=MSB(x) (in case rotR p1, without p2) p1 := repeat (p2) rotR p1 (in case rotR p1, p2)
影响标志位:	C 0 S Z (of the last step)
字节:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=none) 1+p2 (p1=ACCU, p2=NUMBER)
描述:	向右旋转 p1 -->向右移动 p1 寄存器, 将进位寄存器的位给MSB, LSB 被放在进位寄存器中。 将p1 寄存器带有进位向右旋转p2 次 -->将p1 寄存器带有进位向右旋转p2次, 在每次旋转中将进位放到MSB 位中, 将LSB 位放到进位中。
类别:	移动 & 旋转

round	近似值
句法:	round p1, p2
参数:	p1 = ACCU [x] p2 = NUMBER [half scale division]
算法:	p1 = round (p1, p2)
影响标志位:	
字节:	7
循环:	子函数调用
描述:	将x中的数值求近似值。根据所配置的‘半个分度’, 在x中存储的数值将会被求近似值 (向下近似 <5, 向上近似>= 5)。
类别:	其它

setC	置位进位标志
句法:	setC
参数:	-
算法:	-
影响标志位:	C 0
字节:	1
循环:	1
描述:	置位进位标志和清除溢出标志
类别:	移动 & 旋转

shiftL	向左移位
句法:	shiftL p1, (p2)
参数:	p1 = ACCU [x, y, z, r] p2 = 4-Bit number or none
算法:	p1 := p1<< 1; carry :=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) shiftL p1 (in case rotL p1, p2)
影响标志位:	C 0 S Z
字节:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=none) 1 + p2 (p1=ACCU, p2=NUMBER)
描述:	将 p1 左移 --> 将 p1 寄存器的内容左移, 将0 放入LSB 位, 将MSB 位放入进位中。 将 p1 左移p2 次 --> 将 p1 寄存器内容左移 p2 次, 在每次移动中将0 放入LSB 位, 将MSB 位放入进位中。
类别:	移动 & 旋转
setLCD	设置 LCD
句法:	setLCD
参数:	-
算法:	-
影响标志位:	-
字节:	1
循环:	子函数调用
描述:	把LCD寄存器61~62的所有位置1 此操作码与 'newlcd '搭配使用, 来显示所有LCD段。 实际上, 此操作是调用ROM中的子程序。编译时会把此命令转换成相应的跳转命令。
类别:	LCD 显示
shiftR	向右移位
句法:	shiftR p1, (p2)
参数:	p1 = ACCU [x, y, z, r] p2 = 4-位常数或无
算法:	p1 := p1>> 1; carry:=MSB(x) (in case rotL p1, without p2) p1 := repeat (p2) shiftL p1 (in case rotL p1, p2)
影响标志位:	C 0 S Z
字节:	1 (p1=ACCU, p2=none) 2 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=none) 1 + p2 (p1=ACCU, p2=NUMBER)
描述:	带符号右移寄存器p1 --> 右移寄存器 p1 内容, MSB 位根据数值是正数还是负数被复制保留。 带符号的将p1 右移p2 次-->将 p1 寄存器内容右移 p2 次, MSB 位根据数值是正数还是负数被复制保留。
类别:	移动 & 旋转

sign	符号
句法:	sign p1
参数:	p1 = ACCU [x, y, z, r]
算法:	$p1 := p1 / p1 $ $p1 := 1 = 0x000001 \quad \text{if } p1 \geq 0$ $p1 := -1 = 0xFFFFF \quad \text{if } p1 < 0$
影响标志位:	S Z
字节:	2
循环:	2
描述:	以2 的补码形式的带符号的被指定的寄存器。 一个正数返回 1, 一个负数返回 -1 零被考虑为正数
类别:	简单算术运算

skip	跳过
句法:	skip p1
参数:	p1 = NUMBER [1, 2, 3]
算法:	PC := PC + bytes of next p1 lines
影响标志位:	
字节:	1
循环:	1 + 跳过的命令
描述:	无条件跳过 p1
类别:	无条件跳转

skipBitC	有条件跳过
句法:	skipBitC p1, p2, p3
参数:	p1 = ACCU [x, y, z, r] p2 = NUMBER[0..23] p2 = NUMBER[1, 2, 3]
算法:	if (寄存器p1 的p2 位 == 0) PC := PC + bytes of next p3 lines
影响标志位:	-
字节:	1
循环:	1 + 跳过的命令
描述:	如果寄存器p1 的p2 位被清零则跳过p3 命令
类别:	按位运算

skipBitS	有条件跳过
句法:	skipBitS p1, p2, p3
参数:	p1 = ACCU [x, y, z, r] p2 = NUMBER[0..23] p2 = NUMBER[1, 2, 3]
算法:	if (寄存器p1 的p2 位 == 1) PC := PC + bytes of next p3 lines
影响标志位:	-
字节:	1
循环:	1 +跳过的命令
描述:	当寄存器p1 的p2 位被置位时跳过p3 命令
类别:	按位运算

skipCarC	进位清零跳过
句法:	skipCarC p1
参数:	p1 = NUMBER [1, 2, 3]
算法:	if (carry == 0) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 +跳过的命令
描述:	如果进位被清零跳过p1 命令
类别:	跳过标志

skipCarS	进位置位跳过
句法:	skipCarS p1
参数:	p1 = NUMBER [1, 2, 3]
算法:	if (carry == 1) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 +跳过的命令
描述:	当进位标志被置位时跳过p1命令
类别:	跳过标志

skipEQ	等于零跳过
句法:	skipEQ p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (notequalzero == 0) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 + 跳过的命令
描述:	如果前面操作的结果等于零时跳过p1 命令
类别:	跳过标志

skipNE	非零跳过
句法:	skipNE p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (notequalzero == 1) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 + 跳过的命令
描述:	如果前面操作的结果不等于零时跳过p1 命令
类别:	跳过标志

skipNeg	负值跳过
句法:	skipNeg p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (signum == 1) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 + 跳过的命令
描述:	当上一次操作的结果小于零时跳过p1 命令
类别:	跳过标志

skipOvrC	溢出清零跳过
句法:	skipOvrC p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (overflow == 0) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1+跳过的命令
描述:	当溢出标志位清零时跳过p1 命令
类别:	跳过标志

skipOvrS	溢出跳过
句法:	skipOvrS p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (overflow == 1) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 +跳过的命令
描述:	跳过p1 命令如果溢出位被置
类别:	跳过标志

skipPos	正值跳过
句法:	skipPos p1
参数:	p1 = NUMBER[1, 2, 3]
算法:	if (signum == 0) PC := PC + bytes of next p1 lines
影响标志位:	–
字节:	1
循环:	1 + 跳过的命令
描述:	跳过p1 命令如果前面的操作结果大于等于0
类别:	跳过标志

spi2lcd	通过SPI传送LCD显示值
句法:	spi2lcd p1,p2
参数:	p1 = ACCU [y] p2 = NUMBER [number of bits to transmit]
算法:	–
影响标志位:	–
字节:	7
循环:	子程序调用
描述:	此操作把accu y中的数据通过SPI接口传送给外部LCD控制器。第二个参数定义被传送的位数。此操作主要在通过SPI对外部LCD驱动进行操作时会用到。 请注意：此操作需要对PS081进行特殊配置并且需要相应操作码才能正常工作。更多细节请参看4.3章节“支持外部LCD驱动”。
类别:	LCD 显示

ssnPulse	发送脉冲到外部LCD控制器的SSN引脚
句法:	ssnPulse
参数:	–
算法:	SSN -> 0 -> 1 -> 1
影响标志位:	–
字节:	3
循环:	
描述:	在SSN管脚（LCD_SEG3...8管脚）产生正脉冲，以逻辑0结束
类别:	其它

ssnSet	设置外部LCD控制器的SSN引脚为高电平
句法:	ssnSet
参数:	–
算法:	SSN -> 1
影响标志位:	–
字节:	3
循环:	
描述:	设置SSN管脚（LCD_SEG3...8管脚）为高电平
类别:	其它

stop	Stop
句法:	stop
参数:	–
算法:	–
影响标志位:	–
字节:	1
循环:	1
描述:	停止时间数字转换器的工作。 时钟产生源被停止，转换器和EEPROM 进入休眠状态。 对于转换的在启动可以通过外部事件如’看门狗定时器’，’外部开关’或者’新的金属应变测量结果’来触发。一般这个操作码是汇编程序上面的最后一句命令。
类别:	其它

sub	减法
句法:	sub p1, p2
参数:	p1 = NUMBER[1, 2, 3] p2 = NUMBER[1, 2, 3] or 24-位常数
算法:	p1 := p2 - p1
影响标志位:	C O S Z
字节:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
循环:	1 (p1=ACCU, p2=ACCU) 4 (p1=ACCU, p2=NUMBER)
描述:	2个寄存器的减法 一个寄存器和一个常数的减法
类别:	简单算术运算

swap	交换
句法:	swap p1, p2
参数:	p1 = ACCU [x, y, r] p2 = ACCU [x, y, r]
算法:	p1 := p2 and p2 := p1
影响标志位:	-
字节:	1
循环:	3
描述:	交换两个寄存器内容 两个寄存器的内容被交换 Not possible with ACCU[z]
类别:	RAM 操作

useEprInit	初始化用户EEPROM单元
句法:	usrEprInit p1, p2
参数:	p1 = EEPROM CELL [0 to 15] p2 = NUMBER [24-bit]
算法:	EEPROM (p1) = p2
影响标志位:	-
字节:	1
循环:	3
描述:	此操作初始化用户可编程EEPROM空间（地址2000~2047）。总共有16个24位可编程EEPROM单元。 请注意：此操作码只用于初始化。从EEPROM用户单元读/写需要用putepr/getepr操作码。
类别:	EEPROM存取指令

6.5 系统Reset, 休眠模式和自动配置

在接收到复位（上电复位或看门狗复位）、测量结束或者在休眠模式中转换计数器结束计数的信号后ALU会被激活。复位优先于后面两项。首先ALU跳转到ROM代码起始单元（地址2048）。复位后无论ALU是否激活，都会进行一次初始检查。

复位后检查epr_pwr_cfg标志以决定是否把自动配置数据从EEPROM中拷贝到RAM。

接下来检查标志位epr_pwr_prg以决定是否执行EEPROM代码（从地址48开始）。在单芯片运行时，这是合理的，并且epr_pwr_cfg应该被置1。但是在前端操作中，这是不允许的，当epr_pwr_cfg = 0时，微处理器就会被停止。

如果ALU不是由复位激活的，TDC单元开始测量或者在睡眠模式下不进行测量，转换计数器开始计数。之后，检查epr_usr_prg标志以决定是否跳转到EEPROM（地址48）。再者，单机操作时，epr_usr_prg 应该为1，在前端操作时epr_usr_prg 应该为0。

在EEPROM代码中，首先应该检查flg_rstpwr标志，以确定是否是有复位命令引起的跳转。如果是由复位引起的，那么推荐进行更详细的检查来确定这个复位是由上电复位、按键还是看门狗中断引起的。

除此之外，通过查看flg_intav0标志可以知道芯片是仍然处在睡眠状态还是在进行应变测量。

最后停止ALU。到此就执行完了一个完整的ALU复位（包括开始标志位）。同时也复位了程序堆栈。只有RAM中的数据保持不变。

6.5.1 上电复位

当给芯片供电时，就是进行上电复位。整个芯片被复位，只有RAM保持不变。

在epr_pwr_prg = 1时，执行EEPROM地址48中的用户自写代码。

6.5.2 看门狗重置

上电复位也可以由看门狗计时器触发。在微处理器已开启四次都未复位时，可通过操作码“clrwdt”来进行看门狗复位。从状态位flg_wdtalt（寄存器22，位17）可判断出看门狗计时器是否溢出。

在epr_pwr_prg = 1时，执行EEPROM地址48中的用户自写代码。

6.5.3 通过Pin 27进行外部复位

在单机运行模式（SPI_ENA = 0）中，也可以通过27脚（SPI_CSN_RST）进行外部上的复位。这可以相当于一个复位按钮。按钮的状态可以在状态位flg_rstssn（寄存器22，位18）中查看。

在epr_pwr_prg = 1时，执行EEPROM地址48中的用户自写代码。

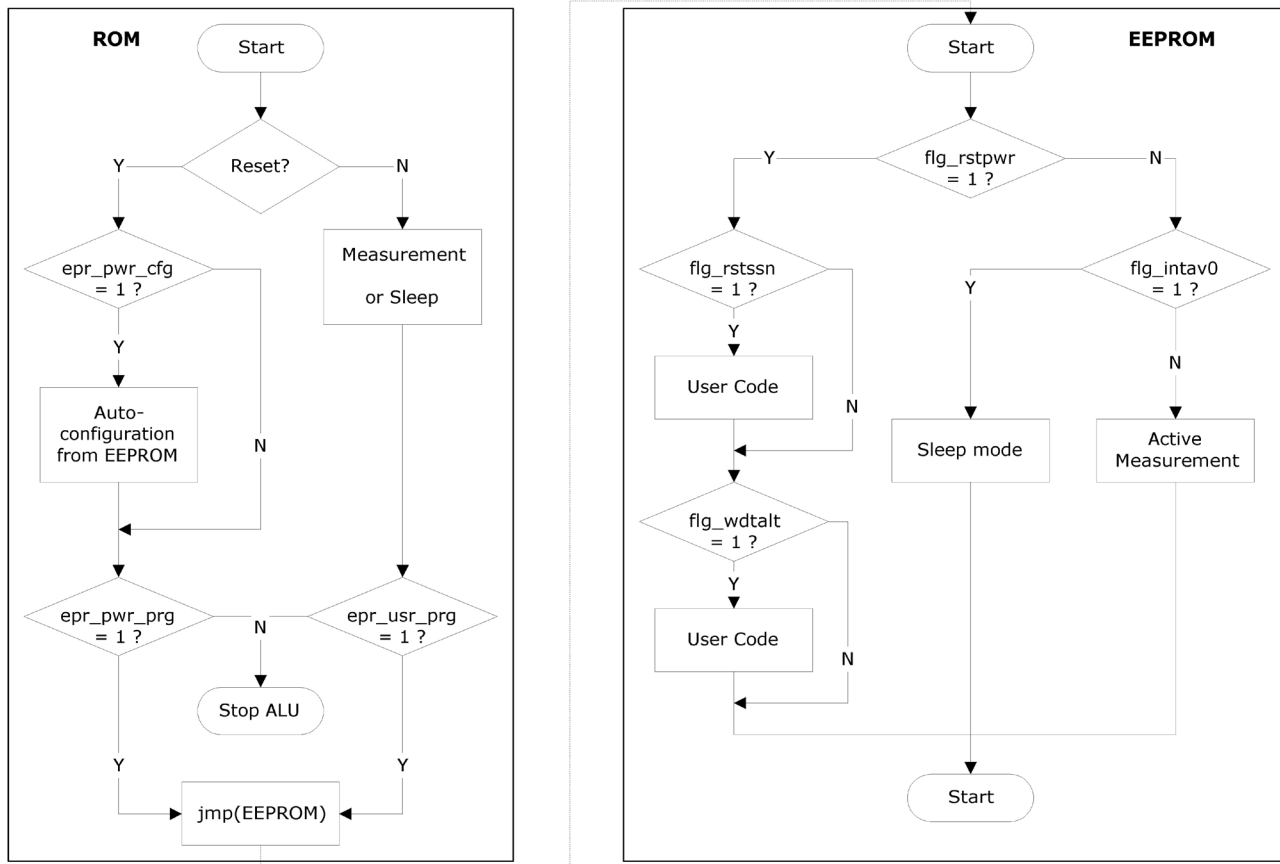


图 6.3 Flow chart

6.5.4 睡眠模式

在睡眠模式下，只有10KHZ的时钟在运行。微处理器定期会被唤醒，但是不进行测量。微处理器在唤醒时检查I/O端口是否有输入。可从状态位flg_intav0（寄存器22，位22）中查看微处理器是否已被从睡眠状态中唤醒。

配置：
 tdc_sleepmode Register 1, Bit 17
 tdc_conv_cnt[11:0] Register 0, Bits 23 到 14

设置tdc_sleepmode = 1进入睡眠模式。此操作与设定avrate = 0作用相同。

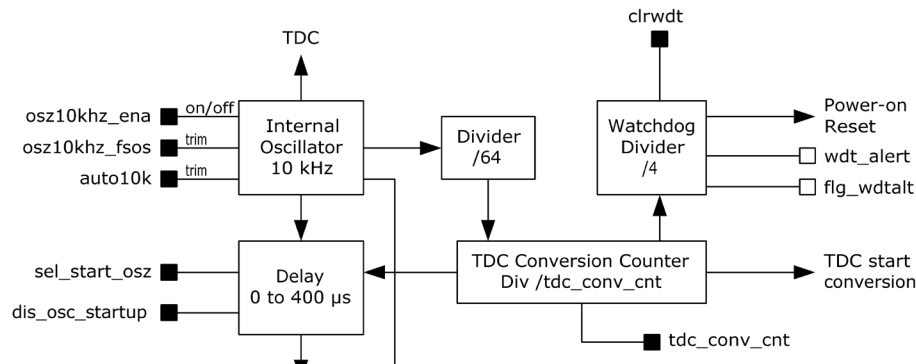
在睡眠模式下，转换计数器tdc_cn timer () 一直运行，直到睡眠模式结束，然后立即开始执行存储在EEPROM中的用户自写程序（从地址48开始）。

从睡眠模式唤醒后，TDC在进行测量之前必须被重新初始化。

6.5.5 CPU 时钟

系统的基础时钟是内部低功耗的10KHZ晶振。它用于：

- 在单一转换模式下触发测量
- 在测量范围2中给TDC单元做前置计数器
- 在拉伸模式下做为周期时间的基准



图片 6.4 时钟产生

6.5.6 看门狗计数器和单次转换计数器

在单一转换模式下TDC转换计数器开启一次测量。它持续运行。单次转换率通过公式 $10\text{kHz} / 64 / \text{tdc\_conv\_cnt}$ 来计算。

随着测量的开始，看门狗的计数器开始增加。看门狗计数器记下转换的次数。在测量结束时，微处理器开始运行用户程序。在正常的操作中，用户的代码结束前发送CLRWDT命令复位看门狗。在TDC出错而未完成测量或是EEPROM代码运行无法结束时，看门狗会引起上电复位。

当通过SPI接口控制PS081时，发送SPI操作码watch_dog_off，可以关闭看门狗。看门狗也可以通过SPI_CSN_RST管脚上的信号沿来复位。

6.5.7 计时器

PS081有一个实时计数器，上电复位后以10ms为周期自动计数。此计数器的实际数值可以在RAM地址29中读取。计数器是24位的，对应的最大计时时间为46小时。可以发送cleartimer操作码复位此计数器。

目录	页面
7 其它.....	7-2
7.1 从 PS08 过渡到 PS081	7-2
7.2 Bug报告	7-3
7.3 文献指导.....	7-3
7.4 文件歷史.....	7-4



7 其它

7.1 从 PS08 过渡到 PS081

下面的表格给出了 PS08和 PS081的不同之处:

	PS08	PS081
EEPROM 程序空间:	1Kx8 bit	2Kx8 bit
EEPROM 用户单元:	2x24 bit	16x24 bit
RAM 用户单元:	32x24 bit	48x24 bit
输入个数:	最大 5	最大 21
GPIOs (Mult_IO) 个数:	1	3
控制寄存器:	17	19
通过SPI驱动外部LCD驱动:	不可以	可以
操作码:		与PS08相同, 进而增加了: - 抗抖动滤波 - 迟滞滤波 - 近似值 - Gramm(克) 到 Stone(磅) 的单位转换
封装:	QFN56	QFN56
裸片:	2770x2520 μm^2 Pad: 116 μm x 90 μm	2370x3470 μm^2 Pad: 116 μm x 90 μm

硬件变化:

从硬件角度上讲, 变化是非常小的. 封装版本应用都是一样的(QFN56), PS08 可以一比一的在现有电路板上被PS081封装片替代. 仅需要注意的是管脚 13 和15的连接变化, PS081的管脚 13 从VCC 变化到--> Mult_IO5, 而管脚 15 从GND 变化到--> Mult_IO4.

那么在现有的PS08 PCB板上用PS081更换, 有两种方式:

在PS081中不连接 13 和15

或者: 连接两个管脚, 将他们配置为输入(configreg_17, en_io4 和 en_io5)

软件变化:

PS08 有17个配置寄存器(configreg_00 到 configreg_16). PS081 多了两个寄存器配置, 另外 configreg_16 也有改变. 也就是你需要配置额外的2个寄存器, 最简单的方法就是选择用寄存器的默认配置即可.

一些操作码有改变, 比如 no2lcd 这个操作码可以有参数设置. 在PS081当中还加入了一些额外的操作码, 例如中值平均滤波(median), 迟滞以及求近似值等操作码. 对于新的操作码的描

述, 您可以在PS081最新版本assembler软件的在线帮助中获得. 对于评估系统软件也有版本的更新.

其他改变:

SPI 操作码 „Read Access “ 被改变. 在PS081当中 4 个EEPROM 单元被按顺序的进行寻址.

如果 AV-rate平均率的设置小于10的话, 参数 „PS081Adjust “ (configreg_03, bit [9:4]) 需要在 PS081当中被设置, PS081adjust需要设置为AV-rate平均率设置的2倍, 和最小为8. 比如AV-rate 为 6, 那么 PS081adjust 需要被设置为12.

7.2 Bug报告

时间	对于Bug所做的改动	解决方案
2009年12月	将全桥的并联模式取消掉 (优化零点偏移) 背景: 这个模式的引入原本是为了通过在芯片上并联 RDSOn电阻降低PS081的零点偏移。但是在我们大量的实验测试过程中并没有显示出好的结果，相反测量的偏移更加严重。因此请应用标准的全桥模式，系统的零点偏移在 $\pm 6\text{nV/V/K}$ 左右，同时可以通过PCB的设计来进行优化。	应用标准的全桥模式
2010年7月	如果有任何 I/O 管脚被用作输出管脚, 那么最多可以配置实现的输入为4个输入. 也就是说, 为了增加输入管脚到21个的话, 那么不能有任何管脚作为输出.	-

7.3 文献指导

数据手册

标题	文本号码	时间
PS081-EVA 测量评估系统 PS081	DB_PS081_EVA V1.0	2009年9月
ALCS-350 V2 高精度应变传感器模拟器	DB_ALCS_V2 V0.1	2009年8月

技术白皮书

标题	文本号码	时间
建立太阳能电池板驱动 电子秤的指导	WP001 V2.0	2008年10月
如何降低传感器的Gain和Offset 偏移 通过PS08的TKGain和TKOffset系 数	WP002 V1.1	2010年2月
Temperature Measurement with PS08	WP003 V1.0	2009年8月
如何应用1 测量原理轻 松实现数字传感器	WP004 V1.0	2010年3月

应用指南

标题	文本号码	时间
Meterological Investigations of PS081 Determining Zero Drift and Gain Drift	AN018 V1.0	2008年7月
Strain Gauge Wiring with PICOSTRAIN	AN012 V1.0	2005年8月
RSPAN-BY-TEMP 温度补偿方法(快速指南)	AN021 V0.5	2009年9月
通过 PS081进行太阳能厨房秤建立的指导	AN022 V1.3	2010年3月
对于太阳能能人体秤设 计的指导-PS081版本	AN023 V1.3	2009年9月
EMI Countermeasures for a Digital Load Cell	AN025 V1.0	2010年6月

所有文档可以在acam公司网站下载:

<http://www.acam.de/download-section/picostrain>

7.4 文件历史

2009年6月22日	First release of preliminary version
2009年7月09日	Second release of preliminary version
2009年9月31日	Release of version V0.3. New Mult_pp factors, new POR circuit, minor corrections
2010年1月15日	Fullbridge parallel mode canceled
2010年2月12日	放松 V0.6, 新的第 3.63, 3.6.4
2010年6月24日	3.4.6, 6.3.1, 7.1

8 索引

目录	页面
9 附录.....	9-2
9.1 高精度研究	9-2
9.2 电路原理图	9-4
9.3 连接驱动外部 LCD driver的代码.....	9-5
9.4 长期的零点偏移的调查研究.....	9-7

9 附录

9.1 高精度研究

为了能够研究这个转换器的最高精度我们选择了HBM的一个高质量的传感器一起来记录测量精度。那么整个配置如下：

- HBM-SP4C3 传感器，连接改为双半桥 Picostrain 连接方式
- 仅应用1个温度补偿电阻Rspan，另一个桥臂的温度补偿电阻被短路
- 电源电压选择为 4.5 V
- PCB为评估系统上的高精度模块
- 应用了Median 滤波

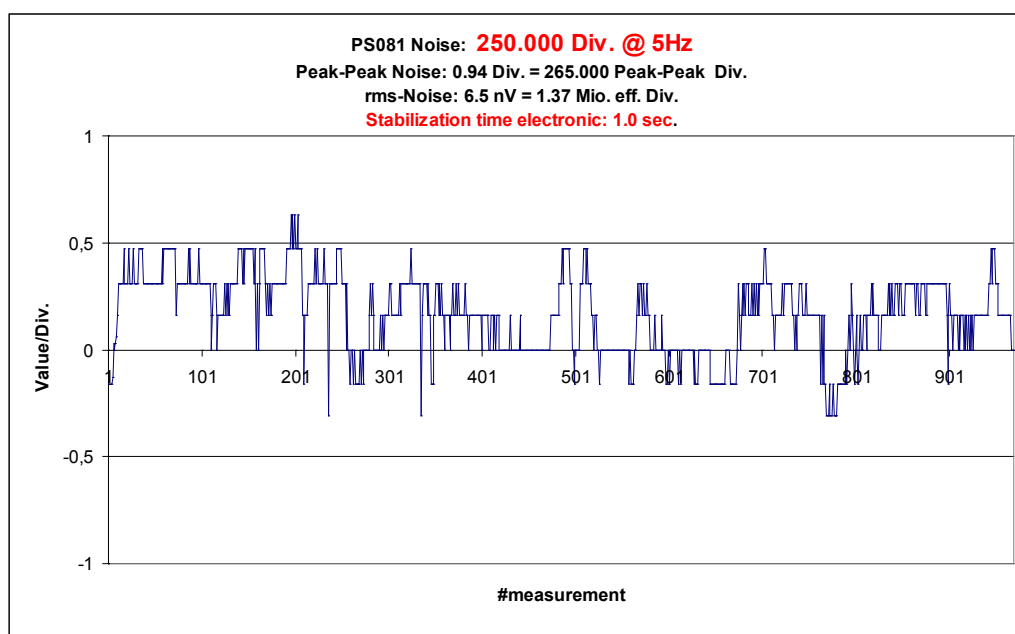
我们获取了两组数据，噪声特性以及响应时间。请参看下面的表格：

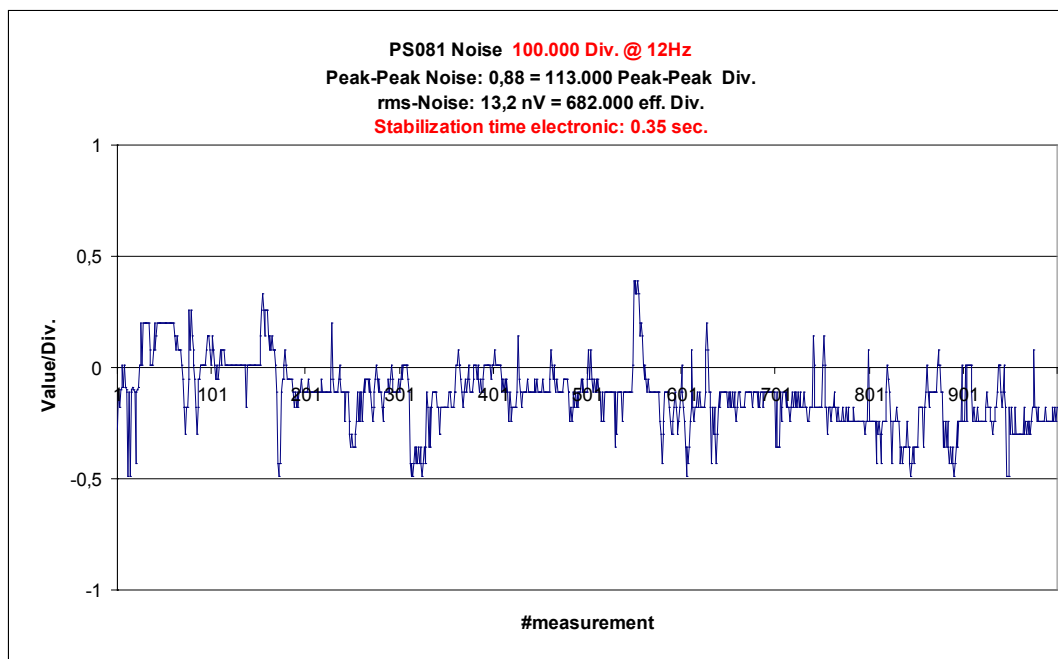
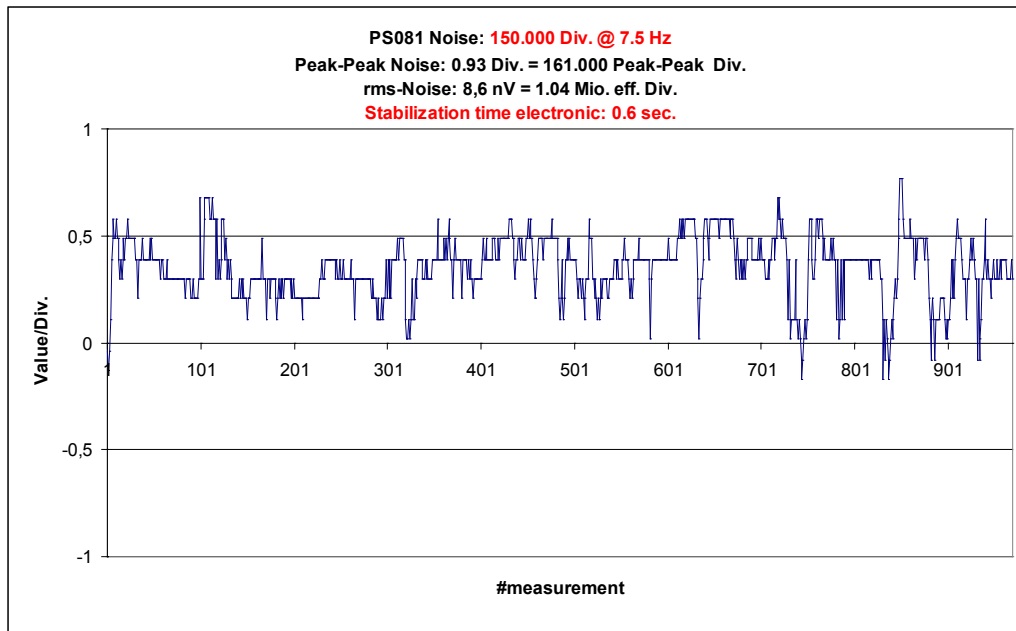
表 8.1 测量精度

测量分度					
峰-峰	内码 (有效.)	有效精度位	输出频率	稳定时间	RMS-噪声 [nV]
250.000	1.370.000	20.3	5 Hz	1.0 sec.	6.5 nV
150.000	1.040.000	19.9	7.5 Hz	0.6 sec.	9 nV
100.000	680.000	19.3	12 Hz	0.35 sec.	13 nV
80.000	595.000	19.1	16.6 Hz	0.27 sec.	15 nV
60.000	385.000	18.5	30 Hz	0.15 sec.	24 nV

下面的3个噪声图表显示了在1个分度内的噪声特性。在图表的描述上您可以看到我们设置的分度值，在所有情况下，噪声都是完全在1个分度之内的。

图 8.1 所达到的精度

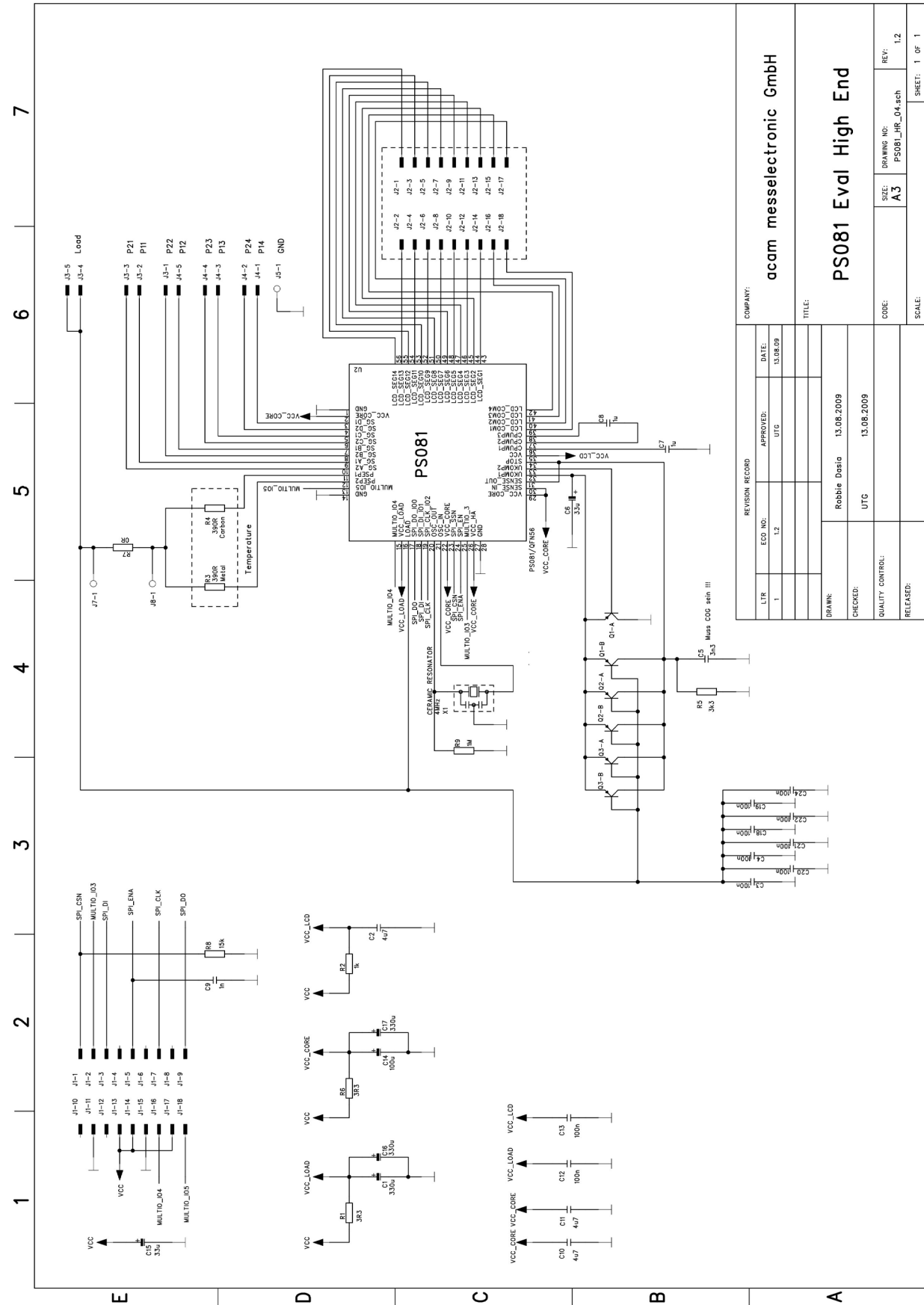




注释:

在这个设置中高精度的实现是需要一个好的传感器保证的，以及软件和硬件设置。如果您需要实现这样高精度的应用，请联系acam公司关于更详细的信息。

9.2 电路原理图



9.3 连接驱动外部 LCD driver的代码

在第四章我们解释了如何连接一个外部的LCD驱动器。下面的程序显示了一个通过SPI接口驱动外部LCD driver的过程。一个简单的例子将会在PS081的汇编assembler软件中给出。

配置:

```

equal 0x480540    ; Config Register 64 ←
equal 0xE00700    ; Config Register 65 ←
equal 0x000000    ; Config Register 66 ←

```

常数定义:

```

CONST spi_chip_select 9
CONST spi_data_out      10
CONST spi_clock         8
CONST lcd_reg_low_read 98
CONST lcd_reg_mid_read 99
CONST spi_bus           65

```

初始化合泰HT1621 (子程序的实现过程):

```

init_holtek:
    ;set initial state
    ramadr 65
    or r, 0x000700    ;set pins lcd_spi_out to 1
    ;set duty cycle and number of common segments (Holtek)
    move y, 0x0004A1
    jsub send_command ;LSB of y is MSB to Holtek!!!
    ;turn on display (Holtek)
    move y, 0x000601
    jsub send_command
    ;turn on bias generator (Holtek)
    move y, 0x000401
    jsub send_command
    ;select crystal oscillator (Holtek)
    move y, 0x000141
    jsub send_command
    jsubret
    ;send command (fixed to 9+3 bits), supporting routine

```

send_command:

```
    ramadr 65
    bitinv r, spi_chip_select
    spi2lcd y, 0x00000C
    ramadr 65
    bitinv r, spi_chip_select
    jsubret
```

发送数据到 Holtek, 1/4MUX LCD:

```
;-----
; 1/4 Multiplex Driver
;-----
;1/4 duty
driver_4:
    ramadr spi_bus
    bitinv r, spi_chip_select
    move y, 0x000005 ;opcode send data for Holtek, set adr 0
    spi2lcd y, 0x000009
    ;first 24 bits
    ramadr lcd_reg_low_read ;reg 98
    move y, r
    spi2lcd y, 0x000018
    ;next 24 bits
    ramadr lcd_reg_mid_read ;reg 99
    move y, r
    spi2lcd y, 0x000018
    ;send last 8 bits, special chars
    ramadr lcd_reg_high
    move y, r
    spi2lcd y, 0x000008
    ramadr spi_bus
    or r, 0x000700 ;set SPI wires back to high
    jsubret
```

调用子函数:

```

ramadr      20      ;HB0 result
move  x, r
no2lcd      x, 1
;----- displaying on external LCD -----
newlcd
jsub        driver_4

```

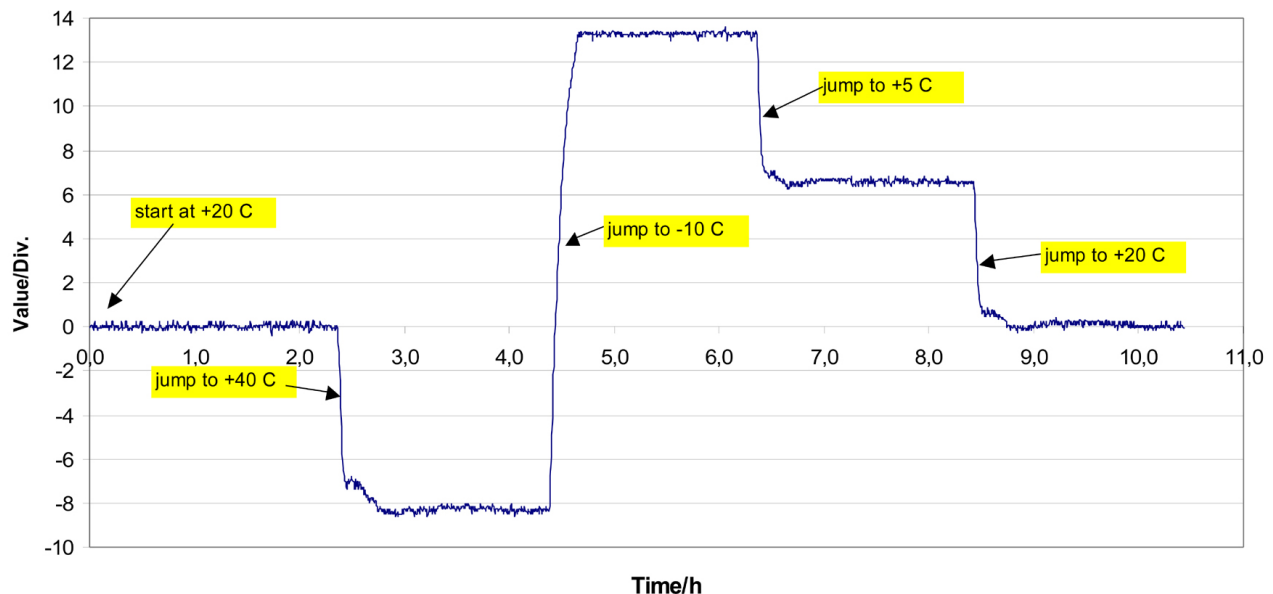
注释:

这个代码显示了通过SPI驱动外接LCD driver的例子。这个并不是一个完整的程序。请确定LCD Driver是正确进行连接的(如第四章所述的将LCD的管脚并联)。

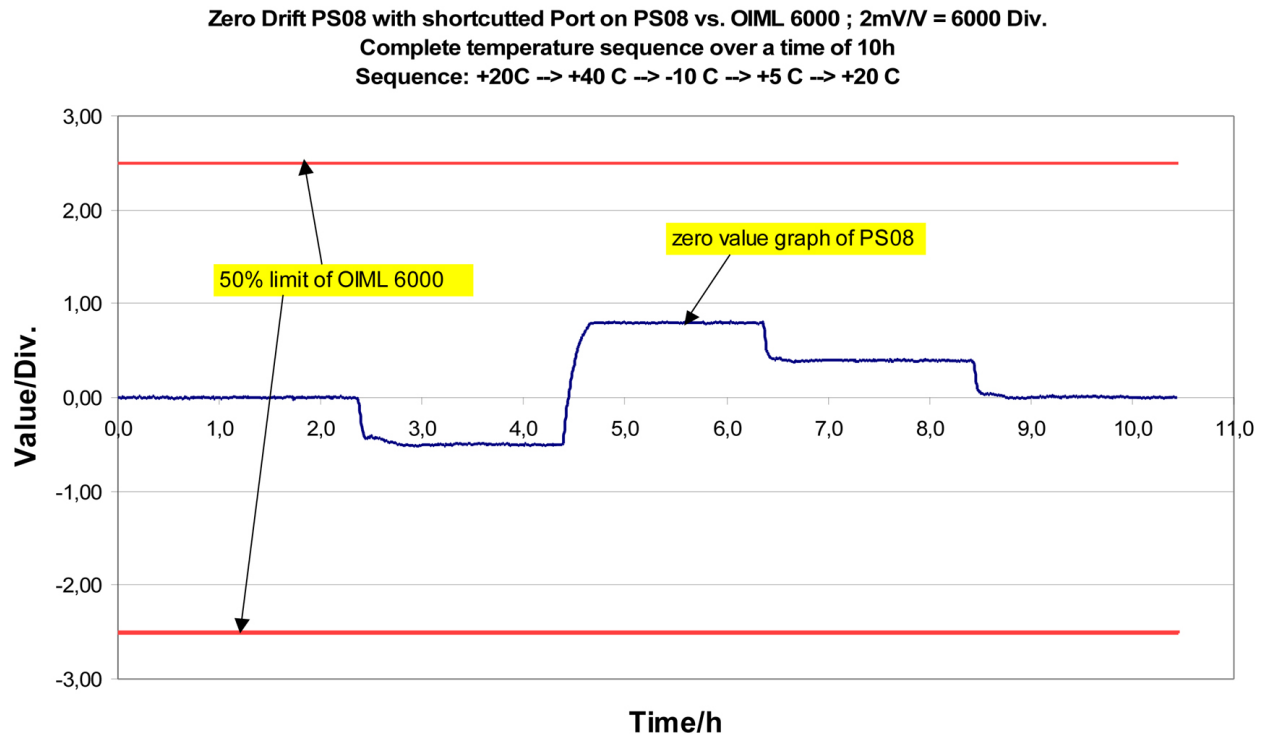
9.4 长期的零点偏移的调查研究

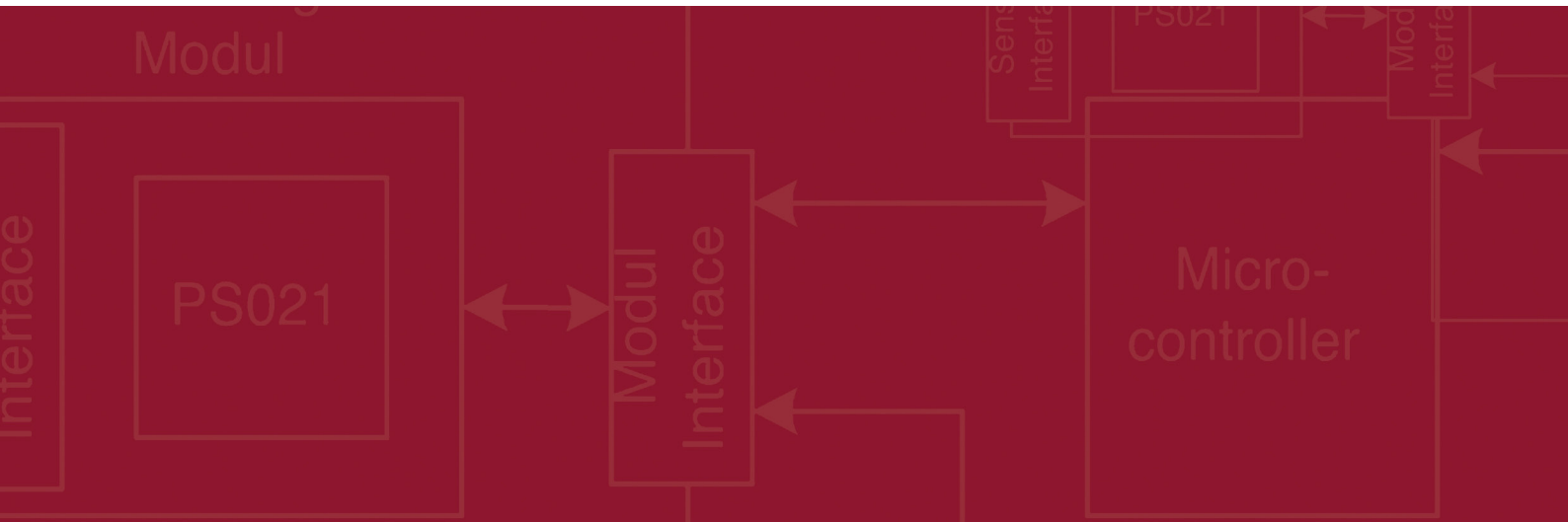
在2009年8月, acam 公司进行了长时间的零点偏移测试调查。在这个测试中, PS08的端口通过固定的电阻短接, 然后将整个电路板放到温度箱内。在10小时的测试当中, 温度循环从 + 20°C→ + 40°C→-10°C→ + 5°C→ + 20°C。那么请参考下面的整个循环的测量曲线。

ZeroDrift PS08 with shortcuted ports on PS08; 2mV/V = 100.000 Div
Temp Sequence: +20 C --> +40 C --> -10 C --> +5 C --> +20 C
Duration of complete run: 10 h



零点偏移是非常低的，完全满足OIML6000的标准，如下图所示：





acam-messelectronic gmbh
Am Hasenbiel 27
76297 Stutensee-Blankenloch
Germany / Allemagne
ph. +49 7244 7419 - 0
fax +49 7244 7419 - 29
e-mail: support@acam.de
www.acam.de